

ESCUELA MILITAR DE INGENIERÍA
“MCAL. ANTONIO JOSÉ DE SUCRE”
BOLIVIA

PROYECTO FINAL



**SISTEMA WEB INTELIGENTE DE GESTIÓN
PREDICTIVA DEL ESTADO OPERATIVO DE LAS
LINEAS DE EMBOTELLADO INTEGRANDO
MODELOS ESTOCÁSTICOS, INTELIGENCIA
ARTIFICIAL Y DISEÑO DE CABLEADO
ESTRUCTURADO DE LA EMPRESA EMBOL S.A.**

**CRUZ SERRANO SHARAID GABRIELA
GUTIÉRREZ LOZANO ELVIN ANDRES
PINTO LUJAN JHERSON ADOLFO
SOSSA CHUGAR THIAGO LEONARDO**

COCHABAMBA, 2025

ÍNDICE DE CONTENIDO



ÍNDICE

CONTENIDO	PÁGINAS
1. INTRODUCCIÓN.....	1
2. ANTECEDENTES	2
3. PLANTEAMIENTO DEL PROBLEMA	4
3.1. IDENTIFICACIÓN DEL PROBLEMA.....	4
3.2. ANÁLISIS CAUSA EFECTO	5
3.3. FORMULACIÓN DEL PROBLEMA.....	6
4. OBJETIVOS	6
4.1. OBJETIVOS GENERAL	6
4.2. OBJETIVOS ESPECIFICOS	6
4.2.1. Objetivos y acciones.....	7
5. JUSTIFICACIÓN	9
6. MARCO TEÓRICO.....	9
6.1. GESTIÓN DEL MANTENIMIENTO	9
6.1.1. Predictivo.....	10
6.1.2. Prescriptivo.....	11
6.2. MODELOS ESTOCÁSTICOS DE DEGRADACIÓN.....	11
6.2.1. Cadenas de Markov	12
6.3. ESTIMACIÓN DE PARÁMETROS Y VALIDACIÓN	12
6.3.1. Testing de un Modelo Entrenado	13
6.4. ANALÍTICA AVANZADA EN MANTENIMIENTO.....	13
6.4.1. Q-Learning y Procesos de Markov	14
6.5. LIBRERÍA	14
6.5.1. Numpy	15
6.5.2. Gymnasium	16
6.5.3. Pandas	16
6.5.4. Stable-baseline3.....	16
6.6. FRAMEWORK.....	17
6.6.1. Django	17
6.7. AGENTE IA	18

6.8.	POSTGRESQL GESTOR DE BASE DE DATOS	18
6.9.	CABLEADO ESTRUCTURADO	19
6.10.	TOPOLOGÍAS DE RED	19
6.10.1.	Topología Estrella.....	19
6.10.2.	Topología árbol.....	20
6.11.	VIRTUAL LOCAL AREA NETWORKS	20
6.12.	PROTOCOLOS DE RED.....	21
6.12.1.	TIA/EIA-568-D	21
6.12.2.	TIA/EIA-1005	22
6.12.3.	WebSocket.....	22
6.13.	INTEGRACIÓN DE DAPHNE COMO SOPORTE PARA WEBSOCKETS	23
7.	INGENIERÍA DEL PROYECTO.....	24
7.1.	ANALIZAR LA INFORMACIÓN TÉCNICA Y OPERATIVA NECESARIA PARA EL MODELADO PREDICTIVO DEL ESTADO DE LOS EQUIPOS, INCLUYENDO VARIABLES DE PROCESO EXTRAÍDAS DE LOS PLC, REGISTROS HISTÓRICOS DE MANTENIMIENTO Y DATOS DE CALIDAD DE PRODUCCIÓN.	24
7.1.1.	Diagrama de clases.....	27
7.1.2.	Diagrama Entidad-Relación (DER).....	27
7.1.3.	Modelo Entidad-Relación extendido (ER).....	28
7.2.	DISEÑAR E IMPLEMENTAR UN MODELO PROBABILÍSTICO BASADO EN CADENAS DE MÁRKOV QUE REPRESENTE LAS TRANSICIONES DE ESTADO DE LOS EQUIPOS Y SIRVA COMO BASE PARA LA PREDICCIÓN DEL COMPORTAMIENTO OPERATIVO EN LAS LÍNEAS DE EMBOTELLADO.....	28
7.3.	DESARROLLAR Y ENTRENAR UN MÓDULO DE INTELIGENCIA ARTIFICIAL QUE COMPLEMENTE EL MODELO ESTOCÁSTICO, PERMITIENDO AJUSTAR LAS PREDICCIONES ANTE COMPORTAMIENTOS NO LINEALES Y CONDICIONES OPERATIVAS CAMBIANTES	32
7.3.1.	Seleccionar variables de entrada relevantes para el entrenamiento	32
7.3.2.	Diseñar e implementar un módulo para el registro de variables de entrada, incorporando mecanismos de validación y persistencia.....	32
7.3.3.	Acciones	34
7.3.4.	Entrenar el modelo de IA.....	35
7.4.	GENERAR REPORTES AUTOMATIZADOS DEL ESTADO DE LOS EQUIPOS EN FUNCIÓN DE LA INFORMACIÓN REGISTRADA.....	38

7.5.	DISEÑAR LA ARQUITECTURA DE RED Y CONSTRUIR EL MODELO DE CABLEADO ESTRUCTURADO EMPLEADO EN LA PLANTA.....	39
7.5.1.	División de subredes	40
7.5.2.	Cableado estructurado	41
7.6.	DESARROLLAR UN SERVIDOR BÁSICO PARA ALOJAR, EJECUTAR Y RESPALDAR EL SISTEMA DE MONITOREO Y CONTROL BASADO EN MODELOS DE INTELIGENCIA ARTIFICIAL Y CADENAS DE MÁRKOV.	45
7.6.1.	Definir las especificaciones esenciales para implementar un servidor básico que soporte la ejecución y almacenamiento del sistema desarrollado.....	45
7.6.2.	Seleccionar la arquitectura de software y hardware adecuada para la implementación del servidor (local o virtualizado).	45
7.6.3.	Configurar el entorno del servidor con los servicios necesario.....	47
7.6.4.	Probar la estabilidad, conectividad y rendimiento del servidor con la carga real del sistema.	48
7.7.	REALIZAR PRUEBAS FUNCIONALES AL SISTEMA COMPLETO INTEGRADO	49
8.	CONCLUSIONES Y RECOMENDACIONES.	50
8.1.	CONCLUSIONES.....	50
8.2.	RECOMENDACIONES.	51
9.	BIBLIOGRAFÍA.....	52

ANEXOS

ÍNDICE DE FIGURAS

CONTENIDO	PÁGINAS
Figura 1. Árbol de problemas	5
Figura 2. Capacidad de producción vs Estado operativo por línea	26
Figura 3. Diagrama de clases.....	27
Figura 4. Diagrama entidad relación	27
Figura 5. Modelo entidad - relación	28
Figura 6. Módulo de Markov.....	30
Figura 7: Interfaz del sistema con optimización con cadenas de Markov.....	31
Figura 8: Modulo de registro de variables	33
Figura 9: Modulo de entrenamiento del agente	35
Figura 10: Imagen del resultado del agente entrenado	36
Figura 11: Resultados del modelo de simulación completo	37
Figura 12: Interfaz de reporte del monitoreo de la Linea K-128	38
Figura 13. Mapa de todas las sucursales de Embol S.A simulado en Packet Tracer	39
Figura 14. Ping a embol.com.....	40
Figura 15. Croquis de la sucursal de LA PAZ.....	41
Figura 16. Croquis de la sucursal de TARIJA.....	42
Figura 17. Croquis de las sucursales de SANTA CRUZ y COCHABAMBA	42
Figura 18: Diagrama de la división del cableado estructurado de cada sucursal.....	43
Figura 19: Imagen del cableado estructurado realizado en la planta de Santa Cruz	44
Figura 20: Creación del servidor con especificaciones	45
Figura 21: Inicio de instalación de la máquina virtual	46
Figura 22: Final de la instalación de la máquina virtual.....	46

Figura 23: Transferencia del sistema comprimido en un zip	47
Figura 24: Interfaz de ingreso al sistema desde el servidor	48
Figura 25: Datos de condiciones simuladas.....	49
Figura 26: Imagen de la compilación e integración de los resultados del sistema	50

ÍNDICE DE ECUACIONES

CONTENIDO	PÁGINAS
Ecuación 1 Ecucación de markov	12
Ecuación 2. Matriz de transición de estados	29
Ecuación 3. Sumatoria de cada fila	29

ÍNDICE DE TABLAS

CONTENIDO	PÁGINAS
Tabla 1. Tabla de objetivo-acción de los objetivos específicos	7
Tabla 2. Tabla de líneas de producción	24
Tabla 3. Tabla de datos específicos de cada línea de producción de EMBOL	25

ÍNDICE DE ANEXOS

CONTENIDO	PÁGINAS
Anexo A: REUNIÓN VIRTUAL (20/05/25).....	1
Anexo B. REUNIÓN VIRTUAL (24/05/25).....	1
ANEXO C: OPERACIÓN AND Y SUBDIVISIÓN DE LA RED	2

CUERPO DEL PROYECTO



1. INTRODUCCIÓN

En la industria de bebidas carbonatadas, donde los volúmenes de producción son altos y los márgenes operativos cada vez más estrechos, las interrupciones inesperadas en las líneas de embotellado representan una amenaza directa a la eficiencia, la rentabilidad y la capacidad de respuesta al mercado. La presión por cumplir entregas obliga a las plantas a mantener una disponibilidad técnica casi continua, donde cualquier minuto de parada implica pérdidas de producto, consumo adicional de insumos y riesgo de incumplimiento de pedidos.

La empresa EMBOL S.A., embotelladora oficial de Coca-Cola en Bolivia, ha logrado altos estándares de calidad bajo certificaciones como ISO 9001, ISO 14001 e ISO 45001. Sin embargo, sus líneas de embotellado aún presentan paradas imprevistas derivadas de una gestión fragmentada de la información crítica de planta. Actualmente, los datos de producción se capturan desde sistemas SCADA, los registros de mantenimiento se documentan manualmente y los resultados de calidad se gestionan por separado, lo que impide obtener una visión integrada del estado real de los equipos. Esta desconexión de datos limita la capacidad para anticipar fallas, deteriora la planificación del mantenimiento y da lugar a intervenciones mal programadas, mermas productivas y sobrecostos operativos.

Ante esta problemática, el presente proyecto propone el desarrollo de un sistema web inteligente que integre la información operativa proveniente de los PLC, los registros de calidad y los reportes de mantenimiento, para predecir de forma precisa el estado de las máquinas. Esta solución combinará modelos estocásticos, como las Cadenas de Markov, con algoritmos de inteligencia artificial, permitiendo pronosticar con mayor exactitud las probabilidades de transición entre estados operativos a lo largo del tiempo. Además, se incluirá el diseño de un modelo de cableado estructurado que respalde la infraestructura tecnológica necesaria para asegurar la conectividad, trazabilidad y estabilidad del sistema.

Con esta herramienta, EMBOL podrá evolucionar desde un mantenimiento rígido por calendario hacia un esquema predictivo basado en condiciones reales y riesgo, optimizando el uso de repuestos, reduciendo tiempos de inactividad y mejorando significativamente su eficiencia operativa.

2. ANTECEDENTES

La Embotelladora Boliviana EMBOL S.A “Coca-Cola”, franquicia oficial de The Coca-Cola Company, opera desde 1972 se encuentra ubicada en el parque industrial “Ramón Darío Gutiérrez” de la ciudad de Santa Cruz de la sierra, en el manzano 6 (P.I -6) sobre el cuarto anillo de circunvalación, entre las avenida paragua y Canal Cotoca en el Parque Industrial de Santa Cruz. Con un total de 240,000.00m² de infraestructura.

Concentra siete líneas de embotellado (KHS 60, Carballo-40, Bardi, Meyer, K-54, Combi-Sidel 135 y K-128). Estas líneas integran sopladora, llenadora, etiquetadora, inspección electrónica y paletizado automático, alcanzando, en conjunto, ritmos que superan las 60 000 botellas por hora en picos de demanda.

El control diario de cada línea se apoya en PLCs (Controladores Lógicos Programables) que registran caudal, presión y temperatura; los datos se visualizan en pantallas HMI (Human Machine Interface) y se exportan al sistema SCADA en intervalos de quince minutos. Paralelamente, el laboratorio de calidad retira botellas para medir torque de tapas, Brix y volumen de CO₂; los resultados se introducen manualmente, si alguno sale de rango, se emite una orden de mantenimiento correctivo que detiene la línea hasta volver a especificación.

El departamento de mantenimiento aplica un programa TPM (Trusted Platform Module) que combina inspección autónoma (limpieza, ajuste y lubricación ejecutados por los propios operarios) con rutinas preventivas basadas en horas de operación: lubricación diaria, cambio de juntas de las válvulas de llenado cada dos semanas e inspección termográfica mensual de tableros eléctricos. Las intervenciones se

registran en hojas de cálculo por línea y se consolidan al final del turno para actualizar los indicadores de disponibilidad y OEE (Overall Equipment Effectiveness) exigidos por los estándares KORE de Coca-Cola.

Aun con certificaciones ISO 9001, ISO 14001 e ISO 45001, el sistema presenta limitaciones: la información de producción, calidad y mantenimiento permanece dispersa (SCADA, LIMS, hojas Excel); algunas paradas menores se anotan a mano, lo que distorsiona la estimación real del tiempo de inactividad; y el plan de mantenimiento sigue siendo esencialmente preventivo, sin un modelo probabilístico que anticipa la transición de los equipos desde el estado operativo hacia fallas, reparación o fuera de servicio. (Baldivieso, 2016).

Esta fragmentación de datos críticos impide detectar a tiempo cuándo una máquina pasará de operación normal a parada correctiva, lo que genera intervenciones tardías, reemplazos prematuros o fuera de tiempo, y un mayor consumo de energía y materias primas. Las paradas inesperadas en líneas de alta velocidad como la Combi-Sidel 135 o la K-128 provocan retrasos en los despachos, mermas productivas y sobrecostos de mantenimiento que afectan directamente la rentabilidad. Además, al carecer de un enfoque basado en condiciones reales, el mantenimiento se programa en función de horas de servicio fijas, lo que incrementa la incertidumbre, complica la logística de repuestos y pone en riesgo el cumplimiento de los niveles de servicio exigidos por el mercado.

A ello se suma que la operación automatizada de la planta con sensores, PLCs, sistemas de inspección electrónica, paneles de control, laboratorios conectados y estaciones de monitoreo depende críticamente de una infraestructura de red estable y eficiente. Sin embargo, esta infraestructura no está formalizada bajo un diseño de cableado estructurado, lo que dificulta la trazabilidad, el mantenimiento y la expansión ordenada del sistema. La ausencia de una arquitectura de cableado documentada y optimizada puede generar fallos de comunicación, ralentización en la transmisión de datos entre dispositivos, y retrasos en la identificación de averías eléctricas o de red. Por ello, la implementación de un diseño de cableado estructurado se plantea como

una necesidad estratégica para garantizar la confiabilidad de las conexiones, optimizar el flujo de información y facilitar futuras integraciones tecnológicas en el entorno de planta.

3. PLANTEAMIENTO DEL PROBLEMA

A continuación, se describe el planteamiento del problema y el análisis causa efecto.

3.1. IDENTIFICACIÓN DEL PROBLEMA

La empresa EMBOL S.A. enfrenta a limitaciones estructurales que afectan su eficiencia operativa. Una de las principales dificultades radica en que la información crítica de las líneas de embotellado, como los datos de producción obtenidos desde el sistema SCADA, los resultados de calidad y los registros de mantenimiento, se gestiona de forma separada y no se encuentra integrada en una única plataforma. Esta falta de integración impide contar con una visión completa y actualizada del estado de los equipos, lo que dificulta anticipar con precisión el momento en que una máquina puede pasar de funcionamiento normal a un estado de falla o reparación.

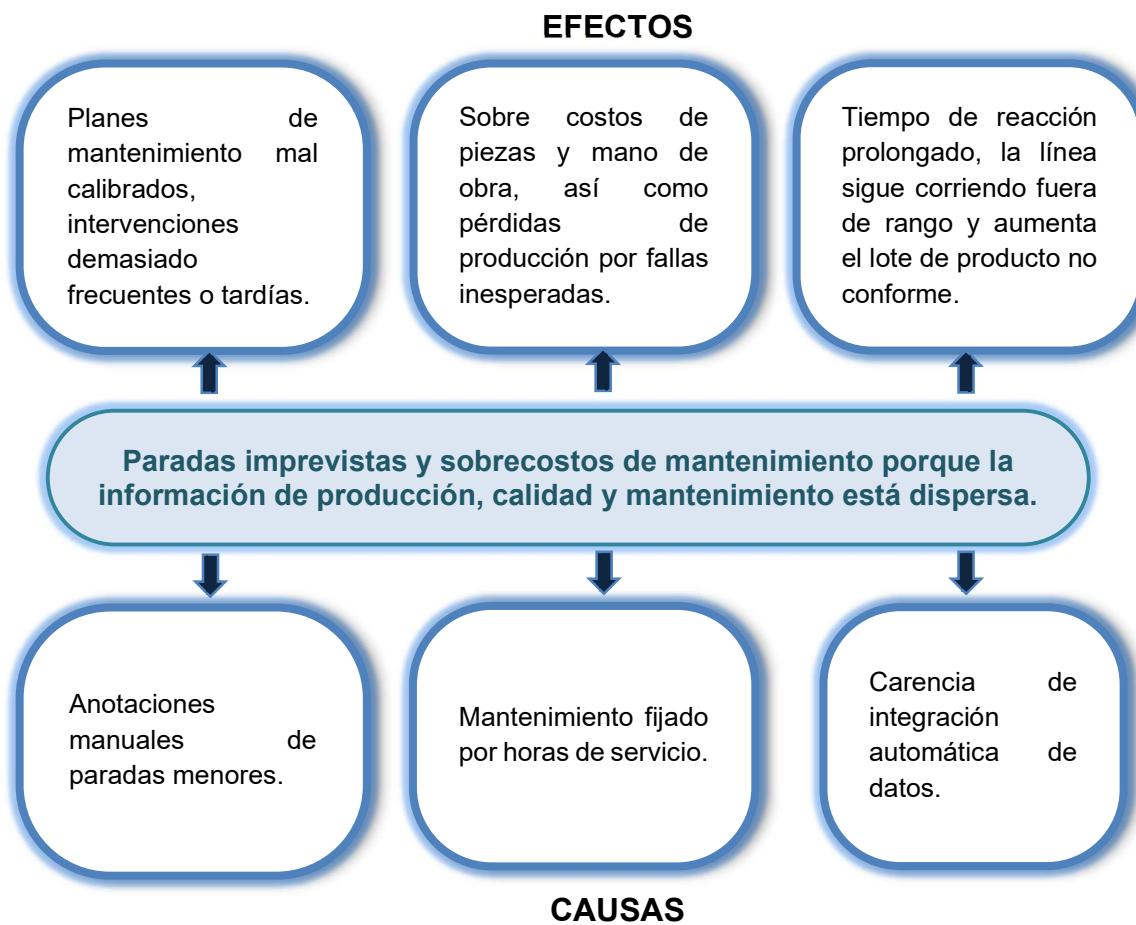
A esto se suma que algunas paradas menores aún se registran manualmente, lo que distorsiona los tiempos reales de inactividad y complica la planificación del mantenimiento. El enfoque actual continúa siendo mayormente preventivo, basado en horas de operación, sin considerar el desgaste real de los componentes ni apoyarse en modelos probabilísticos. Esto genera intervenciones tardías o prematuras, reemplazos innecesarios y un mayor consumo de energía y materias primas. En líneas de alta velocidad como la Combi-Sidel 135 o la K-128, estas interrupciones impactan directamente en los tiempos de despacho y sobrecostos que afectan la rentabilidad.

Finalmente, la operación automatizada de la planta depende de una infraestructura de red confiable que permita el flujo continuo de datos entre sensores, PLCs, sistemas de monitoreo y paneles de control. Sin embargo, esta infraestructura no está respaldada por un diseño de cableado estructurado, lo que dificulta el mantenimiento,

limita la trazabilidad y complica futuras ampliaciones tecnológicas. La ausencia de este diseño reduce la estabilidad de las comunicaciones y debilita la capacidad de respuesta técnica frente a fallas.

3.2. ANÁLISIS CAUSA EFECTO

Figura 1. Árbol de problemas



Fuente: Elaboración propia 2025

- Planes de mantenimiento mal calibrados, intervenciones demasiado frecuentes o tardías, en omisión de un registro sistemático frente a anotaciones manuales de paradas menores.

- Sobrecostos de piezas y mano de obra, así como pérdidas de producción por fallas inesperadas, derivados de un enfoque de mantenimiento fijado por horas de servicio.
- Tiempo de reacción prolongado, la línea sigue corriendo fuera de rango y aumenta el lote de producto no conforme, como resultado de sistemas de datos no conectados automáticamente al proceso operativo.

3.3. FORMULACIÓN DEL PROBLEMA

La empresa EMBOL S.A. presenta dificultades en la gestión eficiente del mantenimiento de sus líneas de embotellado, lo que ha derivado en planes de mantenimiento mal calibrados, con intervenciones que se realizan con demasiada frecuencia o demasiado tarde. Esta situación genera sobrecostos en piezas y mano de obra, además de pérdidas en la producción debido a fallas inesperadas que interrumpen el proceso de manera no programada.

4. OBJETIVOS

4.1. OBJETIVOS GENERAL

Desarrollar un sistema web inteligente de gestión predictiva del estado operativo de las líneas de embotellado integrando modelos estocásticos, inteligencia artificial y diseño de cableado estructurado de la empresa Embol s.a., con el fin de anticipar fallas, optimizar el mantenimiento y mejorar la eficiencia operativa.

4.2. OBJETIVOS ESPECIFICOS

- Analizar la información técnica y operativa necesaria para el modelado predictivo del estado de los equipos, incluyendo variables de proceso extraídas de los PLC, registros históricos de mantenimiento y datos de calidad de producción.

- Diseñar e Implementar un modelo probabilístico basado en Cadenas de Márkov que represente las transiciones de estado de los equipos y sirva como base para la predicción del comportamiento operativo en las líneas de embotellado.
- Desarrollar y Entrenar un módulo de inteligencia artificial que complemente el modelo estocástico, permitiendo ajustar las predicciones ante comportamientos no lineales y condiciones operativas cambiantes.
- Generar reportes automatizados del estado de los equipos en función de la información registrada, proporcionando etiquetas de estado actualizadas y alertas para la toma de decisiones de mantenimiento.
- Diseñar la arquitectura de red y construir el modelo de cableado estructurado empleado en la planta, asegurando la conectividad y trazabilidad de los sistemas involucrados en el monitoreo y control de las líneas.
- Desarrollar un servidor básico para alojar, ejecutar y respaldar el sistema de monitoreo y control basado en modelos de inteligencia artificial y cadenas de Márkov.
- Realizar pruebas funcionales al sistema completo integrado.

4.2.1. Objetivos y acciones

Tabla 1. Tabla de objetivo-acción de los objetivos específicos

OBJETIVO ESPECIFICO	ACCIÓN
Analizar la información técnica y operativa necesaria para el modelado predictivo del estado de los equipos, incluyendo variables de proceso extraídas de los PLC, registros históricos de mantenimiento y datos de calidad de producción	Recopilar datos históricos de fallas, mantenimiento y producción.
	Evaluar la calidad y frecuencia de los registros disponibles.
	Desarrollar la arquitectura de datos del modelo, incluyendo el diseño de base de datos, clases y el Diagrama Entidad-Relación (DER), que soporte la representación y gestión de estados, transiciones y simulaciones.

OBJETIVO ESPECIFICO	ACCIÓN
Diseñar e implementar un modelo probabilístico basado en Cadenas de Markov.	Establecer los estados operativos relevantes.
	Construir la matriz de transición a partir de los datos analizados.
	Simular el modelo y validar su comportamiento frente a los registros históricos.
Desarrollar y entrenar un módulo de inteligencia artificial para complementar el modelo estocástico.	Seleccionar variables de entrada relevantes para el entrenamiento.
	Diseñar e implementar un módulo para el registro de variables de entrada, incorporando mecanismos de validación y persistencia.
	Entrenar el modelo de IA
	Validar y comparar el rendimiento frente al modelo base de Márkov.
Generar reportes automatizados del estado de los equipos en función de la información registrada.	Diseñar etiquetas de estado para cada máquina en tiempo real.
	Programar la generación automática de reportes técnicos y de alertas.
Diseñar la arquitectura de red y construir el modelo de cableado estructurado empleado en la planta.	Levantar el mapa actual de conexiones de red entre equipos de monitoreo y control.
	Diseñar una topología de red estructurada.
	Documentar e ilustrar el modelo final de cableado estructurado en Packet Tracer.
Desarrollar un servidor básico para alojar, ejecutar y respaldar el sistema completo.	Definir las especificaciones esenciales para implementar un servidor básico que soporte la ejecución y almacenamiento del sistema desarrollado.
	Seleccionar la arquitectura de software y hardware adecuada para la implementación del servidor (local o virtualizado).
	Configurar el entorno del servidor con los servicios necesario.
	Probar la estabilidad, conectividad y rendimiento del servidor con la carga real del sistema.

OBJETIVO ESPECIFICO	ACCIÓN
Realizar pruebas funcionales al sistema completo integrado.	Definir pruebas que integren tanto el modelo de Márkov como el módulo de IA, considerando condiciones simuladas.
	Ejecutar pruebas de integración para verificar el funcionamiento individual de cada módulo y su interacción dentro del sistema completo.
	Documentar detalladamente los resultados obtenidos en las pruebas funcionales.

Fuente: Elaboración propia 2025

5. JUSTIFICACIÓN

El presente proyecto responde a una necesidad crítica en la planta de EMBOL S.A.: disminuir las paradas inesperadas en por lo menos seis de las siete líneas operadas en la planta específicamente en las líneas Meyer, Bardi, KHS 60, K-54, Combi-Sidel y K-128, para aprovechar mejor los recursos y tomar decisiones de mantenimiento más precisas. La integración de datos de producción, calidad y mantenimiento en un sistema inteligente permitirá generar alertas oportunas, reportes automatizados y predicciones confiables sobre el estado de cada máquina. Esto facilitará el paso de un mantenimiento fijo por calendario a uno basado en condiciones reales, reduciendo costos de repuestos, liberando mano de obra y acortando los tiempos de inactividad. Como resultado, se incrementará la eficiencia operativa. Además, con un cableado estructurado planificado se garantizará una infraestructura sólida, trazable y escalable. En conjunto, EMBOL avanzará hacia un modelo de mantenimiento predictivo, rentable y alineado con los principios de la Industria 4.0.

6. MARCO TEÓRICO

6.1. GESTIÓN DEL MANTENIMIENTO

La gestión del mantenimiento engloba la planificación, organización y control sistemáticos de todas las tareas necesarias para conservar en condiciones óptimas los activos de una organización. Su objetivo último es evitar paradas no programadas,

maximizar la tasa de producción y asegurar la satisfacción del cliente mediante estrategias que van desde la corrección de fallos hasta la prevención y la predicción de éstos.

Bajo esta visión, el responsable de mantenimiento debe seleccionar y combinar distintos enfoques (correctivo, preventivo, predictivo, TPM, RCM o basado en riesgos) según la criticidad de cada activo, el presupuesto y las metas corporativas. Una correcta matriz de criticidad y el soporte de un sistema CMMS permiten asignar recursos donde el impacto sea mayor, reducir costos generales y optimizar la disponibilidad operacional.

La disciplina ha evolucionado hacia el Mantenimiento 4.0, integrando IoT, big data e inteligencia artificial para monitorizar equipos en tiempo real y predecir fallas con antelación. Esta cuarta “revolución” traslada la toma de decisiones desde simples calendarios hacia analítica avanzada y modelos predictivos, posicionando la gestión del mantenimiento como un pilar de la transformación digital industrial (Pinzón, 2020)

Predictivo

El mantenimiento predictivo se basa en medir y monitorizar variables físicas (temperatura, vibración, corriente, lubricación, etc.) para anticipar el punto futuro de falla y reemplazar componentes justo antes de que ocurra la avería, reduciendo el downtime y alargando la vida útil de los activos. Normalmente emplea técnicas no invasivas y apoyos de instrumentación IoT, análisis estadístico de tendencias y software tipo CMMS-Edge que genera alarmas y órdenes de trabajo en tiempo real. (Pinzón, 2020)

En el contexto de la Industria 4.0, el mantenimiento predictivo se reconoce como una de las principales palancas de digitalización: las empresas que invierten en sensorización, analítica avanzada e IA reportan mejoras de productividad, aunque siguen enfrentando barreras de costo, retorno de inversión y falta de datos integrados. Estudios del Observatorio de Industria 4.0 muestran que solo el 27 % de las compañías

lleva tiempo aplicándolo y que el IoT y el análisis de datos concentran el 42 % de las inversiones tecnológicas actuales. (FUJITSU, 2023)

6.1.2. Prescriptivo

El mantenimiento prescriptivo (RxM) es la evolución del predictivo: no solo detecta condiciones anómalas, sino que genera recomendaciones automáticas sobre qué acción tomar y cuándo, combinando grandes volúmenes de datos OT/IT con algoritmos de análisis avanzado, propios de la Cuarta Revolución Industrial. Integrado en programas de confiabilidad, RxM identifica fallas, prescribe la solución y vincula la ejecución a la estrategia operativa de la planta. (González, 2024)

Las organizaciones que dan el paso desde lo predictivo a lo prescriptivo aprovechan la analítica prescriptiva para optimizar automáticamente la planificación del mantenimiento, minimizar costos y alinear las tareas con los objetivos de negocio; soluciones como Digital Annealer ilustran cómo la optimización en tiempo real puede reducir aún más los gastos operativos. (FUJITSU, 2023)

6.2. MODELOS ESTOCÁSTICOS DE DEGRADACIÓN

Los apuntes Modelos Estocásticos muestran cómo el deterioro acumulativo de un activo puede modelarse como un proceso aleatorio cuya evolución depende de incrementos sucesivos independientes. Un ejemplo típico es la sucesión, donde cada representa la variación de estado entre inspecciones; la cadena resultante cumple la propiedad de Markov y su matriz de transición queda determinada por las probabilidades de esos incrementos, si sólo se permiten saltos positivos, la matriz es triangular superior, capturando el hecho de que la condición del equipo sólo puede igualarse o empeorar con el tiempo. Este enfoque permite calcular, entre otras cosas, la distribución de tiempos hasta alcanzar un nivel crítico de daño y las probabilidades de que la degradación sobrepase ciertos umbrales. (Joaquín Ortega Sánchez, Víctor Rivero Mercado, 2020)

6.2.1. Cadenas de Markov

Una cadena de Markov a tiempo discreto es una sucesión de variables aleatorias que toman valores en un espacio de estados numerable y satisfacen:

Ecuación 1 Ecuación de Markov

$$P(X_{n+1} = j | X_0 = i_0, \dots, X_n = i_n) = P(X_{n+1} = j | X_n = i_n)$$

Es decir, el futuro depende sólo del presente y no del pasado reuniendo las probabilidades, en la matriz de transición P , se dispone de una herramienta compacta para estudiar el comportamiento del sistema a cualquier número de pasos, gracias a las ecuaciones de Chapman-Kolmogorov. En el contexto de degradación, la cadena mencionada antes constituye un ejemplo ilustrativo: las probabilidades generan una matriz con forma escalonada que describe la dinámica de “acumulación de daño”. Con esta representación pueden analizarse tiempos de absorción (llegada al estado “fallo”), recurrencia o transitoriedad de niveles de daño y la existencia de distribuciones estacionarias asociadas. (Joaquín Ortega Sánchez, Víctor Rivero Mercado, 2020)

6.3. ESTIMACIÓN DE PARÁMETROS Y VALIDACIÓN

La fase de estimación arranca con la calibración del modelo, es decir, la obtención de los valores de sus parámetros mediante medición directa o técnicas de optimización (por ejemplo, ajustes de regresión que minimizan una función objetivo). Durante la validación se comprueba el grado de ajuste entre salidas simuladas y observadas con métricas como R^2 , la eficiencia de Nash-Sutcliffe, el índice de Wilmott o la razón RMSE/MAE, elegidas según la sensibilidad a errores constantes, proporcionales o extremos. La evaluación se completa con un análisis sistemático de errores (aproximación, cómputo, propagación) y de la incertidumbre inherente a los datos y al propio modelo, que puede requerir simulaciones Monte Carlo para estimar la propagación de errores cuando el esquema analítico resulta inviable. (Universidad de Murcia, 2021)

6.3.1. Testing de un Modelo Entrenado

Una vez que un modelo de inteligencia artificial ha sido entrenado, es esencial validar si realmente ha aprendido de forma efectiva o si solo ha memorizado los datos que se le dieron. Esta validación se realiza a través de la fase conocida como testing, donde se prueba el modelo con un conjunto de datos totalmente nuevo. El propósito es evaluar su capacidad de generalización, es decir, qué tan bien puede resolver casos que no ha visto antes. Esta etapa permite identificar errores, ajustar parámetros y, sobre todo, asegurar que el modelo es confiable para su uso en entornos reales.

El testing se apoya en distintas métricas para medir el rendimiento del modelo, como la precisión (accuracy), la sensibilidad (recall), la exactitud (F1-score), o el error cuadrático medio, dependiendo del tipo de tarea que se esté resolviendo. Además, ayuda a detectar problemas comunes como el sobreajuste (overfitting), que ocurre cuando el modelo se ajusta demasiado a los datos de entrenamiento y falla con datos nuevos. En contextos industriales o médicos, donde las decisiones del modelo pueden tener un gran impacto, un buen testing no es opcional: es una etapa crítica para garantizar la calidad del sistema. Por ello, se recomienda siempre dividir los datos en tres subconjuntos: entrenamiento, validación y prueba, y evaluar cada uno cuidadosamente antes de poner el modelo en producción. (Géron, 2019)

6.4. ANALÍTICA AVANZADA EN MANTENIMIENTO

La analítica avanzada integra Big Data, Machine Learning y flujos OT/IT para transformar datos en recomendaciones prescriptivas. El modelo RxM (Reliability-centred Maintenance) parte de una fase off-line donde se preparan datos históricos (SCADA, PLC, RCM, FMECA), se construyen modelos estadísticos o de aprendizaje no supervisado y se validan con especialistas en confiabilidad. Una vez implantados on-line, los algoritmos reciben datos en streaming, detectan desviaciones, ejecutan análisis de causa raíz y generan avisos prescriptivos o work-flows automáticos; el sistema incluye re-entrenamiento continuo para adaptarse a cambios operativos. Entre las técnicas empleadas destacan clustering, PCA, redes neuronales, árboles de

decisión, reglas secuenciales y optimización, combinadas con visualizaciones KPI en tiempo real para apoyar la toma de decisiones del personal de mantenimiento. (González, 2024)

6.4.1. Q-Learning y Procesos de Markov

Q-Learning es un algoritmo del aprendizaje por refuerzo, una rama de la inteligencia artificial que se enfoca en enseñar a los sistemas a tomar decisiones a partir de la experiencia. En lugar de depender de datos etiquetados como en el aprendizaje supervisado, aquí el agente explora su entorno, prueba diferentes acciones y aprende de las consecuencias. El objetivo es encontrar una política que maximice la recompensa acumulada a lo largo del tiempo. Lo más interesante es que Q-Learning no necesita conocer cómo funciona el entorno (modelo), lo que lo hace muy útil en situaciones donde el comportamiento del sistema es complejo o desconocido.

Este algoritmo se basa en los llamados Procesos de Decisión de Markov (MDP), que suponen que el resultado de una acción depende únicamente del estado actual del sistema, y no de cómo se llegó a él. Esta propiedad de “memoria limitada” permite que el algoritmo sea más eficiente y directo en su implementación. Q-Learning ha demostrado ser útil en una variedad de escenarios reales: desde robots que aprenden a moverse por sí solos, hasta sistemas que recomiendan productos o servicios personalizados a los usuarios. También se ha utilizado en juegos donde el agente aprende a ganar sin necesidad de programación explícita. Con el tiempo y con suficientes intentos, el agente puede actuar de manera casi óptima, adaptándose a las condiciones del entorno. (Watkins C. &, 1992)

6.5. LIBRERÍA

Una librería de programación es una colección de código desarrollado previamente que los programadores pueden utilizar para desarrollar software de manera más ágil. Estas colecciones de código reutilizable suelen resolver problemas o necesidades comunes de desarrollo. Al usar librerías, el desarrollador no tiene que construir todo desde cero y pueden utilizar las funcionalidades de estas para construir su software.

Sin ellas, el desarrollo de software sería realmente lento e ineficiente. (Rioja U. I., 2023)

6.5.1. Numpy

NumPy es una biblioteca de código abierto para cálculos matemáticos y científicos para tareas de programación en Python. NumPy es la abreviatura de Numerical Python (Python numérico). La biblioteca NumPy ofrece una colección de funciones matemáticas de alto nivel, incluyendo compatibilidad con matrices multidimensionales, matrices enmascaradas y matrices. NumPy también incluye diversas funciones lógicas y matemáticas para dichas matrices, como manipulación de formas, ordenamiento, selección, álgebra lineal, operaciones estadísticas, generación de números aleatorios y transformadas discretas de Fourier.

Python proporciona una estructura de datos básica llamada lista. Una lista de Python admite una secuencia ordenada, modificable o mutable, de elementos de datos o valores llamados elementos. Una sola lista también puede contener muchos tipos de datos diferentes. Esto las hace útiles para almacenar múltiples elementos de datos como una sola variable, como la información de contacto de clientes y los números de cuenta. Sin embargo, las listas son potencialmente ineficientes, ya que consumen una cantidad considerable de memoria y plantean problemas al intentar procesar operaciones matemáticas con diversos tipos de elementos.

En comparación, NumPy se basa en la idea de una matriz de datos homogénea. Si bien una matriz de NumPy puede especificar y admitir varios tipos de datos, cualquier matriz creada en NumPy debe usar solo un tipo de dato deseado; se puede crear una matriz diferente para cada tipo de dato. Este enfoque requiere menos memoria y permite un rendimiento más eficiente del sistema al procesar operaciones matemáticas con elementos de la matriz. (Bigelow, 2024)

6.5.2. Gymnasium

Gymnasium es una biblioteca de Python de código abierto mantenida por la Fundación Farama. Ofrece una amplia colección de entornos prediseñados para agentes de aprendizaje por refuerzo, una API estándar para la comunicación entre algoritmos de aprendizaje y entornos, y un conjunto estándar de entornos compatibles con dicha API. Gymnasium es una bifurcación de OpenAI Gym , que fue lanzado originalmente por OpenAI en 2016. Sin embargo, OpenAI ha cambiado su enfoque y se creó la Fundación Farama para estandarizar y mantener las bibliotecas de RL a largo plazo, incluido Gymnasium. (Carnes, 2023)

6.5.3. Pandas

La librería de código abierto Pandas está específicamente diseñada para la manipulación y análisis de datos en el entorno del lenguaje de programación Python. Se destaca por su potencia, flexibilidad y facilidad de uso, además Pandas facilita el proceso de cargar, alinear, manipular e incluso fusionar datos utilizando Python. Su rendimiento es notable, especialmente cuando el código fuente del back-end está escrito en C o Python. El nombre "Pandas" es una contracción de "Panel Data", referente a series de datos que abarcan observaciones a lo largo de varios períodos temporales. La creación de esta biblioteca se orientó a proporcionar una herramienta de alto nivel para el análisis de datos en Python. Los creadores de Pandas tienen la visión de que esta librería evolucione hasta convertirse en la herramienta de análisis y manipulación de datos de código abierto más poderosa y adaptable, independientemente del lenguaje de programación. (Ferreira, 2022)

6.5.4. Stable-baseline3

Stable Baselines3 (SB3) es una biblioteca de Python que ofrece implementaciones fiables de algoritmos de aprendizaje por refuerzo (RL) en PyTorch. Proporciona una interfaz sencilla y clara para acceder a algoritmos RL de última generación, lo que facilita la investigación, el desarrollo y el uso práctico de técnicas de RL. SB3 se basa en la experiencia del mantenimiento de Stable Baselines (SB2), una bifurcación de

OpenAI Baselines basada en TensorFlow 1.x. Stable-Baselines (SB2) es una biblioteca confiable que ya se ha utilizado en numerosos proyectos y artículos, con más de 300 citas. (Raffin, 2021)

6.6. FRAMEWORK

En informática y programación, un marco proporciona una estructura sobre la que se pueden crear nuevos programas y aplicaciones de software. Un marco puede referirse a un conjunto de funciones de un sistema y su interrelación; a las capas de un sistema operativo o un subsistema de aplicación; o a cómo debe estandarizarse la comunicación en algún nivel de una red. Los frameworks más fiables y populares incluyen diversas herramientas para el desarrollo, la prueba y la depuración de código. Muchos frameworks también proporcionan plantillas que pueden reutilizarse y modificarse según los requisitos de la aplicación. Estos elementos prediseñados y modificables permiten a los programadores crear nuevos programas sin tener que empezar desde cero. Con el framework como base, pueden añadir funcionalidades de alto nivel para crear un producto de software de alta calidad con mayor rapidez y menos errores. (Raul Awati, Ben Lutkevich, 2024)

6.6.1. Django

Django es un framework eficiente para la mayoría de las aplicaciones web y móviles , pero puede presentar dificultades al gestionar múltiples solicitudes simultáneas. Normalmente, los desarrolladores solucionan esta limitación utilizando servidores de aplicaciones como Gunicorn o uWSGI para ejecutar múltiples trabajadores que puedan analizar estas solicitudes. Además, al ser un framework monolítico, las diversas funciones de Django pueden resultar excesivas para proyectos pequeños. Los principiantes que implementan el framework podrían encontrar confuso el número de componentes disponibles. (Doyle, 2024)

6.7. AGENTE IA

Un agente de inteligencia artificial (IA) se refiere a un sistema o programa que es capaz de realizar tareas de forma autónoma en nombre de un usuario u otro sistema diseñando su flujo de trabajo y utilizando las herramientas disponibles.

Los agentes de IA pueden abarcar una amplia gama de funcionalidades más allá del procesamiento del lenguaje natural, incluida la toma de decisiones, la resolución de problemas, la interacción con entornos externos y la ejecución de acciones.

Estos agentes se pueden implementar en diversas aplicaciones para resolver tareas complejas en diversos contextos empresariales, desde el diseño de software y la automatización de TI hasta herramientas de generación de código y asistentes conversacionales. Utilizan las técnicas avanzadas de procesamiento del lenguaje natural de los modelos de lenguaje de gran tamaño (LLM) para comprender y responder a las entradas de los usuarios paso a paso y determinar cuándo recurrir a herramientas externas. (Gutoswka, 2024)

6.8. POSTGRESQL GESTOR DE BASE DE DATOS

Según (Fuentes, 2012) una base de datos es un conjunto de datos que pertenecen al mismo contexto, almacenados sistemáticamente para su posterior uso, es una colección de datos estructurados según un modelo que refleje las relaciones y restricciones existentes en el mundo real. Los datos que han de ser compartidos por diferentes usuarios y aplicaciones deben mantenerse independientes de éstas, y su definición y descripción han de ser únicas estando almacenadas junto a los mismos.

PostgreSQL es un sistema de gestión de base de datos relacional de código abierto que permite a los usuarios almacenar, recuperar y gestionar datos de forma segura y eficiente. Es conocido por su rendimiento robusto, flexibilidad y capacidad de escalabilidad, lo que lo convierte en una opción popular para aplicaciones web y empresariales.

6.9. CABLEADO ESTRUCTURADO

El cableado estructurado es un sistema de infraestructura de telecomunicaciones que organiza y distribuye el tendido de cables en un edificio o instalación de manera ordenada, estandarizada y eficiente. Está diseñado para soportar múltiples servicios como voz, datos, video, control y automatización, independientemente del fabricante o del tipo de equipo que se conecte posteriormente. Esta estructura se basa en normas internacionales (como ANSI/TIA/EIA-568) que definen cómo deben instalarse los componentes físicos (cables, conectores, paneles, etc.) para asegurar compatibilidad, rendimiento y facilidad de mantenimiento.

Su importancia radica en que proporciona una base sólida, flexible y escalable para el funcionamiento de una red de comunicaciones moderna. Al estar estandarizado, el cableado estructurado permite realizar cambios, reparaciones o ampliaciones sin afectar todo el sistema, lo cual ahorra tiempo y reduce costos operativos. Además, una instalación bien planificada mejora la estética, reduce interferencias y facilita la identificación de fallos, optimizando así el soporte técnico. En el contexto actual de digitalización y convergencia tecnológica, contar con un sistema de cableado estructurado eficiente es clave para asegurar la continuidad del servicio, el rendimiento de la red y la adaptación a futuras necesidades de conectividad. (Rioja R. M., 2025)

6.10. TOPOLOGÍAS DE RED

La topología de red se refiere a la forma en que los nodos y las conexiones están dispuestos física y lógicamente en una red. Las redes constan de una serie de enlaces y nodos. Los nodos incluyen dispositivos como enruteadores, comutadores, repetidores y ordenadores. (Jackson, 2024)

6.10.1. Topología Estrella

La topología estrella es una de las configuraciones más utilizadas en redes de cableado estructurado. En este esquema, todos los dispositivos de red (como computadoras, impresoras o cámaras) se conectan directamente a un nodo central,

que generalmente es un switch o un hub. Esta estructura tiene la forma de una estrella, donde cada dispositivo cuenta con su propio cable dedicado hacia el centro.

La principal ventaja de esta topología es su facilidad de administración y diagnóstico. Si un dispositivo o cable presenta fallas, el resto de la red sigue funcionando normalmente, ya que no dependen unos de otros para comunicarse. Esto mejora considerablemente la confiabilidad del sistema. Además, su diseño facilita la localización de problemas, simplifica la ampliación de la red y permite un control más eficiente del tráfico de datos. Por estas razones, la topología estrella es la más recomendada en instalaciones modernas de cableado estructurado, tanto en oficinas como en centros educativos, industrias y edificios inteligentes. (Forouzan, 2002)

6.10.2. Topología árbol

Según (Forouzan, 2002) esta estructura de red se utiliza en aplicaciones de televisión por cable, sobre la cual podrían basarse las futuras estructuras de redes que alcancen los hogares. También se ha utilizado en aplicaciones de redes locales analógicas de banda ancha para así verificar que tan importante sería esta topología en el futuro.

6.11. VIRTUAL LOCAL AREA NETWORKS

Una LAN es un conjunto de elementos físicos y lógicos, los cuales son capaces de proveer interconexión a una gran cantidad de dispositivos de comunicación de información en un área privada restringida; de manera más clara, una LAN es un conjunto de equipos que se encargan de configurar una red de comunicación para que ésta permita la transmisión de datos de un dispositivo a otro, ya sea dentro de una habitación, un edificio o un conjunto de edificios. Las VLANs (Virtual Local Area Networks - Redes Virtuales de Área Local) como su nombre lo indica, son LANs virtuales que representan agrupaciones de trabajo, las cuales se encuentran definidas por software y además mantienen comunicación entre sí, como si estuvieran conectadas a un mismo concentrador; aunque realmente estén localizadas en diferentes segmentos de red pertenecientes a algún edificio. (UNAM, 2021)

6.12. PROTOCOLOS DE RED

Las redes informáticas funcionan bajo un conjunto de reglas denominadas protocolos de red, que son las que establecen cómo y en qué contexto pueden comunicarse entre sí los dispositivos conectados a ellas. Estos protocolos de una red definen los estándares para la transmisión de datos, por lo que son los encargados de garantizar que la información enviada por un dispositivo sea comprendida correctamente por el receptor. (GoDaddy, 2024)

6.12.1. TIA/EIA-568-D

Este estándar provee especificaciones para el diseño de las instalaciones y la infraestructura edilicia necesaria para el cableado de telecomunicaciones en edificios comerciales. Este estándar tiene en cuenta tres conceptos fundamentales relacionados con telecomunicaciones y edificios:

- Los edificios son dinámicos. Durante la existencia de un edificio, las remodelaciones son comunes, y deben ser tenidas en cuentas desde el momento del diseño. Este estándar reconoce que el cambio ocurre y lo tiene en cuenta en sus recomendaciones para el diseño de las canalizaciones de telecomunicaciones.
- Los sistemas de telecomunicaciones son dinámicos. Durante la existencia de un edificio, las tecnologías y los equipos de telecomunicaciones pueden cambiar dramáticamente. Este estándar reconoce este hecho siendo tan independiente como sea posible de proveedores y tecnologías de equipo.
- Telecomunicaciones es más que “voz y datos”. El concepto de Telecomunicaciones también incorpora otros sistemas tales como control ambiental, seguridad, audio, televisión, alarmas y sonido. De hecho, telecomunicaciones incorpora todos los sistemas de “bajo voltaje” que transportan información en los edificios.

Es de fundamental importancia entender que para que un edificio quede exitosamente diseñado, construido y equipado para soportar los requerimientos actuales y futuros de los sistemas de telecomunicaciones, es necesario que el diseño de las telecomunicaciones se incorpore durante la fase preliminar de diseño arquitectónico. (Joskowicz, 2006)

6.12.2. TIA/EIA-1005

La norma ANSI/TIA/EIA 1005-Telecommunication Infrastructure Standard for Industrial Premises, especifica el proyecto y las prácticas de construcción para edificios industriales, detallando los requisitos de cableado, distancias, configuraciones y topologías que suplementan la norma general de Edificaciones Comerciales (EIA/TIA 568C-2). El estándar adoptado en esta norma crea tres niveles de hostilidad de los ambientes industriales (conocidos como MICE), representando el “3” el nivel más crítico:

- M: Mecánico (impacto, vibración, tensión, torsión, etc.).
- I: Ingreso (partículas sólidas y líquidas).
- C: Climático y Químico (temperatura, humedad, radiación solar, productos químicos, etc.).
- •E: Interferencias electromagnéticas (descarga en contacto y en arco, radiofrecuencia, tensión de línea, inducción, etc.). (Sanhueza, 2012)

6.12.3. Websocket

websocket es un protocolo de red que proporciona comunicación full-dúplex a través de una única conexión TCP persistente. A diferencia de HTTP, que sigue un modelo de solicitud-respuesta, los websocket permiten una comunicación continua, bidireccional y de baja latencia entre el cliente y el servidor. La API Websocket es una interfaz basada en navegador que permite a las aplicaciones web establecer una conexión persistente con un servidor. Esto facilita el intercambio de datos en ambas direcciones sin necesidad de repetidas solicitudes HTTP. (Diaconu, 2025)

6.13. INTEGRACIÓN DE DAPHNE COMO SOPORTE PARA WEBSOCKETS

Daphne es un servidor de protocolos HTTP, HTTP2 y Websocket para aplicaciones ASGI (Interfaz de Puerta de Enlace de Servidor Asíncrona). Se utiliza comúnmente como componente de servidor para los Canales de Django, lo que permite que las aplicaciones de Django gestionen comunicaciones en tiempo real, como Websocket y conexiones de larga duración. Daphne está diseñado para funcionar a la perfección con las capacidades asíncronas de Django, lo que lo convierte en una opción potente para crear aplicaciones de chat, notificaciones en vivo y otras funciones en tiempo real. Escrito en Python y basado en el framework de redes Twisted, Daphne gestiona eficientemente múltiples conexiones concurrentes, garantizando un rendimiento estable y escalable.

Como servidor ASGI, Daphne actúa como puente entre los clientes web y las aplicaciones Django, gestionando las solicitudes entrantes y enviándolas a las capas de aplicación correspondientes. Admite su ejecución como parte de una pila de implementación junto con proxies inversos como NGINX o Traefik, lo que ayuda a gestionar la terminación HTTPS y el balanceo de carga. Daphne es especialmente útil para aplicaciones que requieren comunicación bidireccional, como herramientas colaborativas, actualizaciones en tiempo real o juegos en línea. Su integración con los canales de Django lo convierte en una herramienta esencial para los desarrolladores que buscan ampliar las capacidades de Django más allá de las interacciones síncronas tradicionales de solicitud-respuesta. (Bendoraitis, 2025)

7. INGENIERÍA DEL PROYECTO

7.1. ANALIZAR LA INFORMACIÓN TÉCNICA Y OPERATIVA NECESARIA PARA EL MODELADO PREDICTIVO DEL ESTADO DE LOS EQUIPOS, INCLUYENDO VARIABLES DE PROCESO EXTRAÍDAS DE LOS PLC, REGISTROS HISTÓRICOS DE MANTENIMIENTO Y DATOS DE CALIDAD DE PRODUCCIÓN

Para desarrollar un sistema predictivo confiable, se inició con la recopilación estructurada de información operativa proveniente de las diferentes líneas de embotellado de EMBOL S.A. Esta etapa incluyó datos de capacidad de producción, tipo de producto, frecuencia de mantenimiento, condiciones térmicas, estado operativo estimado y nivel de uso de cada línea.

Tabla 2. Tabla de líneas de producción

LÍNEA	CAPACIDAD (BOTELLAS/HORA)
Meyer	9,000
Bardi	960
KHS	8,700 a 9,000
K-54	12,900 a 21,000
Combi-Sidel	19,980 a 28,800
K-128	24,000 a 36,000

Fuente: Elaboración propia 2025

Embol en general utiliza siete líneas de embotellamiento, pero en la planta de Santa Cruz solo utiliza 6 de estas ya que la línea carballo 40 con una capacidad de producción de aproximadamente 9000 a 16200 botellas/hora, fue retirada de operación en la planta, sin que se disponga de una justificación oficial en la documentación revisada. Esta situación deja operativas únicamente seis líneas en la planta de Santa Cruz, cuyas especificaciones técnicas se detallan a continuación en la siguiente tabla:

Tabla 3. Tabla de datos específicos de cada línea de producción de EMBOL

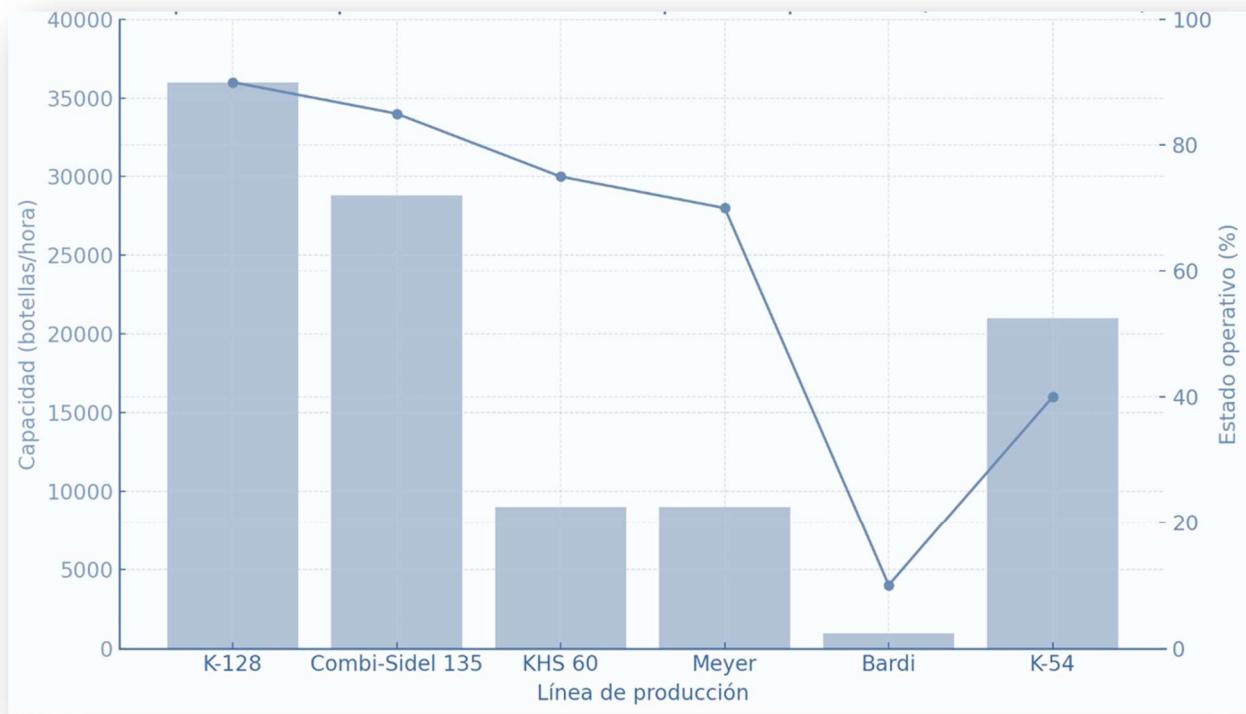
LÍNEA / MÁQUIN A	TIPO DE PRODUCTO	CAPACIDA D (Botella/hora)	ESTADO ACTUAL	FRECUENCIA DE MANTENIMIEN TO	ESTAD O TÉRMIC O (°C)	ESTADO OPERATIV O ESTIMAD O
K-128	PET gaseosa	36,000	Operativa (alta demanda)	Cada 6 meses + sensores online	24-26 °C	90% Uso
Combi-Sidel 135	Multiformato (PET, vidrio)	28,800	Operativa (crítica)	Preventivo mensual	22-25 °C	85% Uso
KHS 60	Gaseosa	9,000	Operativa	Preventivo semestral	25 °C	75% Uso
Meyer	Agua embotellada	9,000	Operativa	Semestral	23-24 °C	70% Uso
Bardi	Vidrio	960	Reserva (no uso diario)	Solo inspección visual	-	10% Uso
K-54	Multisabor PET	21,000	En recuperación	Post-falla	Variab le	40% Uso

Fuente: Elaboración propia 2025

En el siguiente gráfico comparativo evidencia la relación entre la capacidad nominal de producción de cada línea de embotellado y su estado operativo actual. Las líneas **K-128** y **Combi-Sidel 135**, con capacidades de hasta 36,000 y 28,800 botellas por hora respectivamente, mantienen un alto nivel de uso operativo (90 % y 85 %), lo que confirma que son líneas críticas dentro del esquema de producción. En contraste, líneas como **Bardi** y **K-54**, con menor uso efectivo (10 % y 40 %), reflejan potencial de optimización o reserva operativa. Esta visualización respalda la priorización de estas dos líneas principales para el desarrollo y validación del modelo predictivo, ya que su inactividad representa mayor impacto económico y operativo. Asimismo, permite identificar oportunidades de redistribución o mejora de eficiencia en líneas

subutilizadas. Esta información fue esencial para definir los estados del modelo estocástico y los puntos de captura de datos para el sistema web inteligente.

Figura 2. Capacidad de producción vs Estado operativo por línea

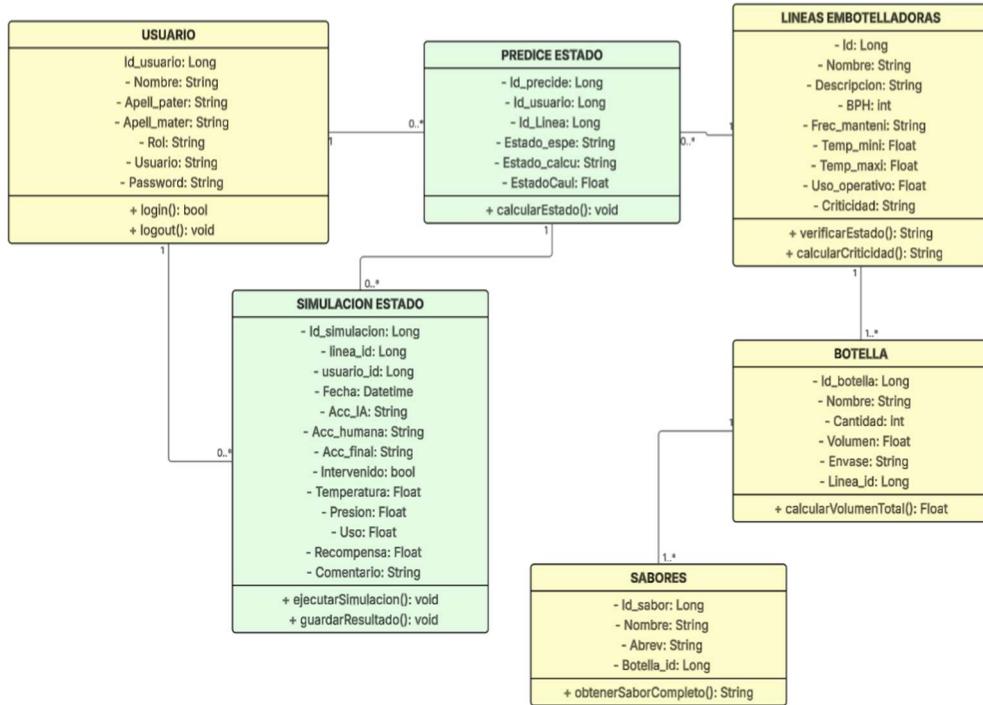


Fuente: Elaboración propia 2025

Teniendo en cuenta la recopilación de datos operativos, así como la estructura y funciones de los módulos a ser desarrollados, se presenta a continuación el análisis estructural del sistema. Este incluye los diagramas de clases, Entidad-Relación (DER) y su versión extendida, los cuales reflejan cómo se organizan, relacionan y gestionan los datos dentro de la plataforma inteligente propuesta. Se recalca que como tal se maneja una estructura para un sitio web de monitoreo de las líneas.

7.1.1. Diagrama de clases

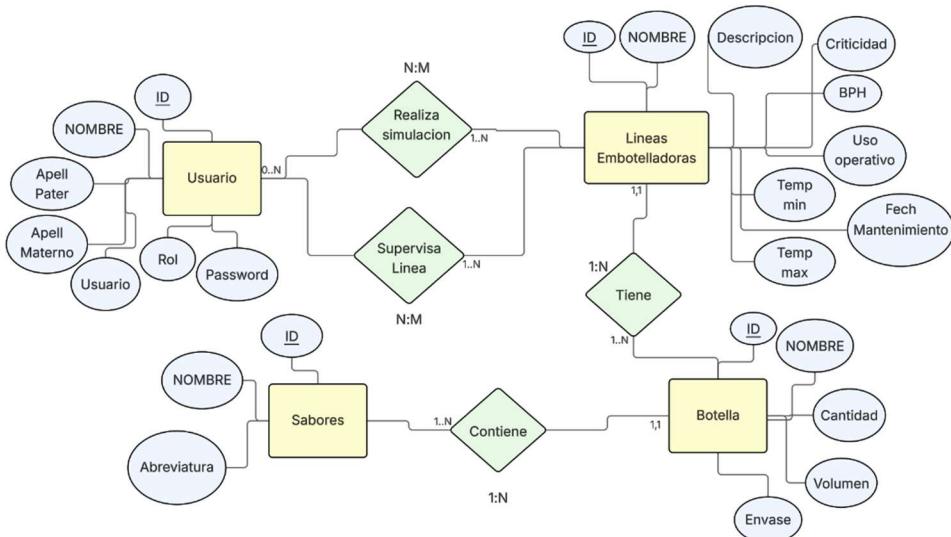
Figura 3. Diagrama de clases



Fuente: Elaboración propia 2025

7.1.2. Diagrama Entidad-Relación (DER)

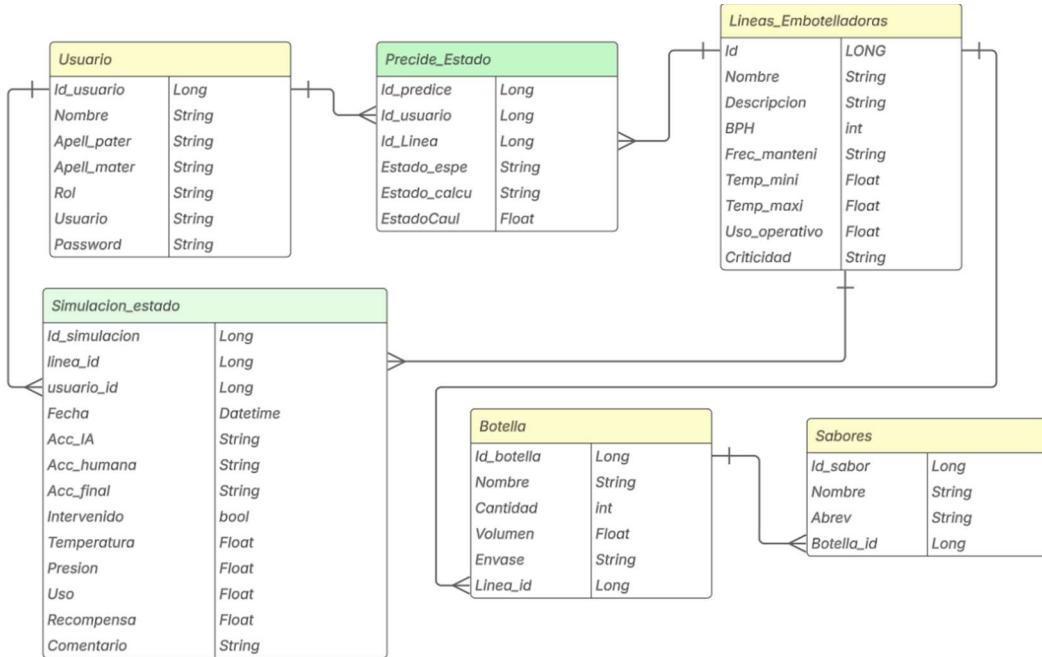
Figura 4. Diagrama entidad relación



Fuente: Elaboración propia 2025

7.1.3. Modelo Entidad-Relación extendido (ER)

Figura 5. Modelo entidad - relación



Fuente: Elaboración propia 2025

7.2. DISEÑAR E IMPLEMENTAR UN MODELO PROBABILÍSTICO BASADO EN CADENAS DE MÁRKOV QUE REPRESENTE LAS TRANSICIONES DE ESTADO DE LOS EQUIPOS Y SIRVA COMO BASE PARA LA PREDICCIÓN DEL COMPORTAMIENTO OPERATIVO EN LAS LÍNEAS DE EMBOTELLADO

Tomando en cuenta la información obtenida de Embol S.A. se tendrá los siguientes estados discretos:

- Operativa (O)
- Mantenimiento (M)
- Recuperación (R)
- Reserva (V)
- Parada (P)

La matriz de transición de Markov para cada línea de producción tendrá esta estructura general:

Ecuación 2. Matriz de transición de estados

$$P = \begin{bmatrix} P_{OO} & P_{OM} & P_{OR} & P_{OV} & P_{OP} \\ P_{MO} & P_{MM} & P_{MR} & P_{MV} & P_{MP} \\ P_{RO} & P_{RM} & P_{RR} & P_{RV} & P_{RP} \\ P_{VO} & P_{VM} & P_{VR} & P_{VV} & P_{VP} \\ P_{PO} & P_{PM} & P_{PR} & P_{PV} & P_{PP} \end{bmatrix}$$

Ecuación 3. Sumatoria de cada fila

$$\sum_{j=J}^5 p_{ij} = 1 \quad \forall i$$

Para determinar las probabilidades de transición, el sistema tomará en cuenta:

- **Temperatura** (desviaciones térmicas pueden indicar deterioro o sobrecarga)
- **Presión** (desajustes indican desbalance o riesgo mecánico)
- **Porcentaje de uso** (carga operativa como variable de fatiga o sobreuso)

Estas variables se integran en un algoritmo de estimación (o red neuronal supervisada) que alimenta la matriz de transición de cada línea.

Estas matrices ya determinadas con datos se llenarán y procesarán en la siguiente clase de Markov.

En la siguiente clase se implementa un modelo probabilístico basado en cadenas de Márkov de tiempo discreto, diseñado para representar y simular el comportamiento de las líneas de embotellado. Esta clase permite validar matrices de transición y vectores de estado, calcular el vector estacionario (estado de equilibrio del sistema), simular la evolución del sistema a través de múltiples transiciones, y transformar datos observacionales en matrices de probabilidad.

Figura 6. Módulo de Markov

The screenshot shows a code editor window with a dark theme. The title bar says "markov.py X" and "Configuración X". The code is written in Python and defines a class named "Markov". The class has several methods: __init__, Verificar, set_matrix, set_vector, resolver_vector_estacionario, Resolver_saltos, and Convertir_Probabilidad. The code uses numpy arrays and includes error handling for invalid matrices and vectors.

```
maquinaria > Markov > markov.py > Markov > __init__.py
1 import numpy as np
2
3 class Markov:
4     def __init__(self, matrix=None, vector=None):
5         self.matrix = np.array(matrix) if matrix is not None else None
6         self.vector = np.array(vector) if vector is not None else None
7
8     if self.matrix is not None and self.vector is not None:
9         if not self.Verificar():
10             raise ValueError("Matriz no válida")
11     def Verificar(self):
12         if self.vector is None or self.matrix is None:
13             return False
14         if not np.isclose(sum(self.vector), 1):
15             return False
16         for i in range(len(self.matrix)):
17             if not np.isclose(sum(self.matrix[i]), 1):
18                 return False
19         return True
20     def set_matrix(self, matrix):
21         self.matrix = np.array(matrix)
22         if self.vector is not None and not self.Verificar():
23             raise ValueError("Matriz no válida al establecer matriz")
24     def set_vector(self, vector):
25         self.vector = np.array(vector)
26         if self.matrix is not None and not self.Verificar():
27             raise ValueError("Vector no válido al establecer vector")
28     def resolver_vector_estacionario(self):
29         n = self.matrix.shape[0]
30         A = self.matrix.T - np.eye(n)
31         A = np.vstack([A, np.ones(n)])
32         b = np.zeros(n + 1)
33         b[-1] = 1
34         solucion = np.linalg.lstsq(A, b, rcond=None)[0]
35         return solucion
36     def Resolver_saltos(self, saltos):
37         if self.matrix is None or self.vector is None:
38             raise ValueError("Matriz o vector no inicializados")
39         new_vector = self.vector
40         for _ in range(saltos):
41             new_vector = np.dot(new_vector, self.matrix)
42         return new_vector
43     def Convertir_Probabilidad(self, Matriz):
44         self.matrix = []
45         for i in range(len(Matriz)):
46             suma = sum(Matriz[i])
47             fila= [Matriz[i][j] / suma for j in range(len(Matriz[i]))]
48             self.matrix.append(np.array(fila))
```

Fuente: Elaboración propia 2025

La clase se representa su uso en la siguiente interfaz web que representa la capa de interacción del usuario con el módulo de simulación de Cadenas de Márkov desarrollado previamente. Desde aquí, el usuario puede seleccionar una máquina específica (por ejemplo, "Línea k-128") para realizar un estudio de comportamiento operativo predictivo. La interfaz despliega el estado actual del equipo (como "Mantenimiento") y presenta los posibles estados del sistema definidos.

Figura 7: Interfaz del sistema con optimización con cadenas de Markov

La captura de pantalla muestra una interfaz web titulada "Optimización con Cadenas de Markov". En la parte superior, hay un menú con opciones como "Markov", "Q-Learning", "Modo Manual", "¡Bienvenido, Equipo Coca-Cola!", y "Cortar pausa". El contenido principal incluye:

- Selección de máquina:** Un cuadro desplegable que muestra "Línea k-128".
- Estado actual:** Se indica "Mantenimiento".
- Estados posibles:** Una lista que incluye "Mantenimiento", "Operativa", "Parada", "Recuperación" y "Reserva".
- Días de simulación:** Un cuadro de texto que contiene "Ej: 5".
- Botón "Aplicar Cambios":** Un botón que permite ejecutar la simulación basada en los cambios introducidos.

Fuente: Elaboración propia 2025

El usuario puede ingresar un número de días de simulación, lo cual actúa como parámetro de entrada para el método `Resolver_saltos` del módulo `Markov.py`. Este método calcula, mediante multiplicaciones sucesivas con la matriz de transición, la probabilidad de que el equipo se encuentre en cada uno de los posibles estados después del número de días indicado. Al presionar "Aplicar Cambios", el sistema ejecuta la simulación estocástica y devuelve el vector de estados probabilísticos futuros, permitiendo visualizar qué tan probable es que el equipo esté en un estado dado dentro del rango de tiempo seleccionado.

7.3. DESARROLLAR Y ENTRENAR UN MÓDULO DE INTELIGENCIA ARTIFICIAL QUE COMPLEMENTE EL MODELO ESTOCÁSTICO, PERMITIENDO AJUSTAR LAS PREDICCIONES ANTE COMPORTAMIENTOS NO LINEALES Y CONDICIONES OPERATIVAS CAMBIANTES

7.3.1. Seleccionar variables de entrada relevantes para el entrenamiento

Para el entrenamiento del agente de aprendizaje por refuerzo en el entorno de simulación de líneas embotelladoras, se seleccionaron como variables de entrada los siguientes parámetros: Uso, Temperatura, Presión y Estado. Estas variables fueron elegidas por su alta relevancia en la dinámica operativa del sistema. El Uso representa el nivel de actividad de la línea, la Temperatura y la Presión reflejan las condiciones físicas del proceso, mientras que el Estado indica la situación operativa actual (como funcionamiento normal, parada o mantenimiento). La selección se basó en criterios de impacto directo en el rendimiento, capacidad de reacción del agente y disponibilidad histórica de datos confiables. Estas variables permiten al modelo capturar de forma representativa el comportamiento del entorno para tomar decisiones óptimas en función de la situación actual.

7.3.2. Diseñar e implementar un módulo para el registro de variables de entrada, incorporando mecanismos de validación y persistencia.

En la figura 8 se muestra el módulo para registro de variables de entrada que permite vincular dinámicamente los datos de cada línea con un entorno simulado que puede registrar variables como temperatura, presión, ciclos de mantenimiento, etc.

generar múltiples entornos (multi_env), se prepara el sistema para alimentar un agente de aprendizaje automático que pueda interactuar, aprender y generalizar comportamientos. Esta infraestructura es esencial para entrenar modelos que luego se validarán y compararán con el modelo base de Márkov.

Figura 8: Modulo de registro de variables

```
def __init__(self):
    self.multi_env = None

    self.nombres_lineas = self._obtener_nombres_lineas()
    self.num_lineas = len(self.nombres_lineas)

Windsurf: Refactor | Explain | Generate Docstring | X
def _obtener_nombres_lineas(self):
    return list(Lineas_Embotelladoras.objects.values_list("Nombre",
        flat=True).order_by('id'))

Windsurf: Refactor | Explain | Generate Docstring | X
def _Construir_Entornos(self):
    Windsurf: Refactor | Explain | Generate Docstring | X
    def make_env(nombre, line_index, total_lines):
        Windsurf: Refactor | Explain | Generate Docstring | X
        def __init__():
            return LineaAdaptativaEnv(nombre, line_index, total_lines)
        return __init__

    envs = [make_env(nombre, i, self.num_lineas)
            for i, nombre in enumerate(self.nombres_lineas)]

    self.multi_env = DummyVecEnv(envs)
    return self.multi_env
```

Fuente: Elaboración propia 2025

7.3.2.1. Agente

En el sistema desarrollado, el agente inteligente encargado de tomar decisiones no se construye manualmente desde cero, sino que se implementa utilizando la librería Stable-Baselines3, mediante el algoritmo de aprendizaje por refuerzo PPO (Proximal Policy Optimization). Este agente es de tipo basado en políticas estocásticas, lo que significa que aprende una política de acción probabilística mediante interacción con el entorno. En cada iteración, el agente observa el estado actual del entorno —representado por variables como el uso de línea, temperatura, presión y estado operativo— y decide una acción, recibiendo una recompensa según su efectividad. El entorno, definido en la clase “LineaAdaptativaEnv”, se adapta dinámicamente a los datos históricos de cada línea embotelladora. La política del agente está representada por una red neuronal que se ajusta progresivamente para maximizar las recompensas acumuladas, permitiendo que el modelo aprenda comportamientos óptimos o eficientes a través de múltiples episodios de entrenamiento. Adicionalmente, el agente

se entrena de forma paralela en múltiples entornos usando “DummyVecEnv”, lo que favorece la generalización del aprendizaje en diferentes líneas de producción.

7.3.2.2. Entorno

El espacio de acciones también está discretizado o continúo dependiendo del diseño interno, y representa las decisiones posibles que el agente puede tomar para ajustar las condiciones de operación (como reducir presión, modificar temperatura o cambiar el régimen de uso). A cada acción, el entorno responde con una recompensa numérica que evalúa si dicha decisión fue beneficiosa en función de los objetivos de operación, como estabilidad del sistema, eficiencia energética o cumplimiento de parámetros de calidad. Además, el entorno avanza temporalmente sobre un conjunto de datos históricos reales simulados (365 días), permitiendo así evaluar decisiones en contextos variados.

El entorno LineaAdaptativaEnv es una implementación personalizada que simula el comportamiento operativo de una línea embotelladora, diseñado específicamente para entrenar agentes de aprendizaje por refuerzo profundo. Desde una perspectiva técnica, este entorno sigue la interfaz de gymnasium.Env, lo cual permite su compatibilidad con algoritmos estándar de RL como PPO. El entorno define un espacio de observación continuo de tipo Box, que incluye cuatro variables clave: nivel de uso (%), temperatura de operación ($^{\circ}\text{C}$), presión interna (bar), y estado codificado de funcionamiento. Estas variables representan la percepción del entorno, es decir, lo que el agente observa en cada paso de tiempo.

7.3.3. Acciones

El sistema actúa como un agente reactivo mejorado que no toma decisiones de optimización directa, pero simula un entorno con base en datos históricos y modelos de transición. Las "acciones" no son de control activo, sino de simulación realista, con intención de que un futuro agente PPO o Q-learning pueda aprender políticas óptimas sobre estos datos.

Las acciones generan salidas o indicadores clave: Secuencia temporal de estados: Histórico simulado de cambios de estado, Serie de valores de uso operativo: Evolución horaria del uso, Serie de temperaturas simuladas: Influida por el horario, estado y uso, Serie de presiones: Calculadas a partir de los valores anteriores.

7.3.4. Entrenar el modelo de IA

Figura 9: Modulo de entrenamiento del agente

```

from Markov.Services.ServicesMarkov import ServicesMarkov
from .ServicesQ_learning import ServicesQ_learning
import numpy as np
from stable_baselines3.common.vec_env import DummyVecEnv

from Entorno import LineaAdaptativaEnv
from stable_baselines3 import PPO
from Reportes.models import Lineas_Embotelladoras
from gymnasium import spaces
class Entrenamiento():
    def __init__(self,Name_linea:str =None):
        self.Name_linea = Name_linea if Name_linea is not None else None

    def set_Linea(self,Name_linea:str):
        self.Name_linea = Name_linea

    def _calcular_espacio_global(self):
        nombres = self._obtener_nombres_lineas()
        temp_min = min(Lineas_Embotelladoras.objects.filter(Nombre=n).values_list("Temp_min", flat=True)[0] for n in nombres)
        temp_max = max(Lineas_Embotelladoras.objects.filter(Nombre=n).values_list("Temp_max", flat=True)[0] for n in nombres)
        pres_base = min(Lineas_Embotelladoras.objects.filter(Nombre=n).values_list("Presion_base", flat=True)[0] for n in nombres)
        pres_max = max(Lineas_Embotelladoras.objects.filter(Nombre=n).values_list("Presion_maxima", flat=True)[0] for n in nombres)

        return spaces.Box(
            low=np.array([0.0, temp_min, pres_base, 0]),
            high=np.array([100.0, temp_max, pres_max, 4]),
            dtype=np.float32
        )

    def _obtener_nombres_lineas(self):
        return list(Lineas_Embotelladoras.objects.values_list("Nombre", flat=True))

    #solo sirve para entrenamiento esto
    def _crear_env_por_linea(self,Name_linea:str =None,obs_space=None):
        q_learning = ServicesQ_learning(Name_linea)
        df = q_learning.Simulacion_Completa(365)
        sensor_data = df[['Uso', 'Temperatura', 'Presion', 'Estado']].values

        return lambda: LineaAdaptativaEnv(Name_linea,
                                           sensor_data,
                                           obs_space)

    def _Construir_Entornos(self):
        nombres_lineas = self._obtener_nombres_lineas()
        obs_space = self._calcular_espacio_global()

```

Fuente: Elaboración propia 2025

La clase de entrenamiento en este sistema tiene como objetivo generar datos simulados que reflejen el comportamiento dinámico de una línea embotelladora bajo distintas condiciones operativas. Para ello, se parte de una configuración real obtenida desde una base de datos, la cual incluye parámetros físicos y operativos clave como:

el estado inicial de la línea, la criticidad, el uso operativo permitido, la temperatura mínima y máxima, así como las presiones base y máxima.

Durante el entrenamiento, se simula una línea de producción a lo largo de un intervalo temporal (por ejemplo, varios días en intervalos horarios). En cada unidad de tiempo, el sistema: Estima el estado operativo actual utilizando una matriz de transición de Markov. Calcula el uso operativo esperado mediante una función logística, ajustada según la criticidad. Simula la temperatura interna considerando efectos cíclicos (modelo de Fourier), el nivel de uso y penalizaciones por estado. Calcula la presión de operación combinando temperatura, uso y el impacto del estado.

Estas variables (estado, uso, temperatura, presión) conforman un conjunto de datos sintéticos que alimentan el entorno de entrenamiento de un agente de aprendizaje por refuerzo (por ejemplo, uno basado en Q-learning o PPO). Estos datos permiten al agente aprender patrones de transición y evaluar políticas sin comprometer un entorno físico real.

Figura 10: Imagen del resultado del agente entrenado

```
In [1]: from q_learning.Services.Entrenamiento import Entrenamiento
In [2]: entrenar = Entrenamiento()
In [3]: entrenar.Entrenar_modelo()
D:\Maquinarias\maquinas\venv\Lib\site-packages\gymnasium\spaces\box.py:235: UserWarning: gym.logger.warn(
gym.logger.warn(
Using cpu device
-----
time/
  fps      | 9643 |
  iterations | 1    |
  time_elapsed | 0   |
  total_timesteps | 1536 |
-----
time/
  fps      | 1969 |
  iterations | 2    |
  time_elapsed | 1   |
  total_timesteps | 3072 |
train/
  approx_kl      | 0.01735357 |
  clip_fraction  | 0.126 |
  clip_range     | 0.2   |
  entropy_loss   | -0.678 |
  explained_variance | -0.0137 |
  learning_rate   | 0.0003 |
  loss            | 42.6  |
  n_updates       | 10    |
  policy_gradient_loss | -0.011 |
```

Fuente: Elaboración propia 2025

El módulo de inteligencia artificial (IA) fue diseñado para complementar el modelo probabilístico del sistema, proporcionando capacidades adaptativas frente a condiciones operativas variables. Este componente emplea un enfoque de aprendizaje por refuerzo, donde un agente interactúa con un entorno simulado que replica el comportamiento de las líneas de embotellado en función de variables críticas como temperatura, presión, vibración y porcentaje de uso.

Figura 11: Resultados del modelo de simulación completo

	In [2]: model = ServicesQ_learning('Línea KHS')
	In [3]: print(model.Simulacion_Completa(2))
	Hora Estado Uso Temperatura Presion
0	0 Recuperación 3.706231 27.409050 29.042605
1	1 Recuperación 0.000000 25.767638 31.828869
2	2 Recuperación 2.260698 27.902466 31.203235
3	3 Recuperación 0.560438 27.278498 28.661263
4	4 Recuperación 0.910882 27.757137 27.563877
5	5 Recuperación 0.000000 27.181852 27.172643
6	6 Recuperación 7.286107 31.735969 32.720173
7	7 Recuperación 4.505213 29.610240 27.427773
8	8 Recuperación 10.575445 33.173077 29.591326
9	9 Recuperación 18.088399 37.671673 34.092957
10	10 Recuperación 22.875690 40.436186 36.190610
11	11 Recuperación 31.984846 46.057788 40.786481
12	12 Recuperación 35.178804 47.866223 42.281187
13	13 Recuperación 37.995759 49.429605 42.062296
14	14 Recuperación 43.317811 52.589590 45.800431
15	15 Recuperación 41.743225 51.218882 44.334128
16	16 Recuperación 44.734275 52.778241 44.245271
17	17 Recuperación 47.016287 53.878735 44.330536
18	18 Recuperación 47.535750 53.898238 43.039489
19	19 Operativa 56.265693 54.390849 58.518979
20	20 Operativa 54.839460 53.480585 57.128765

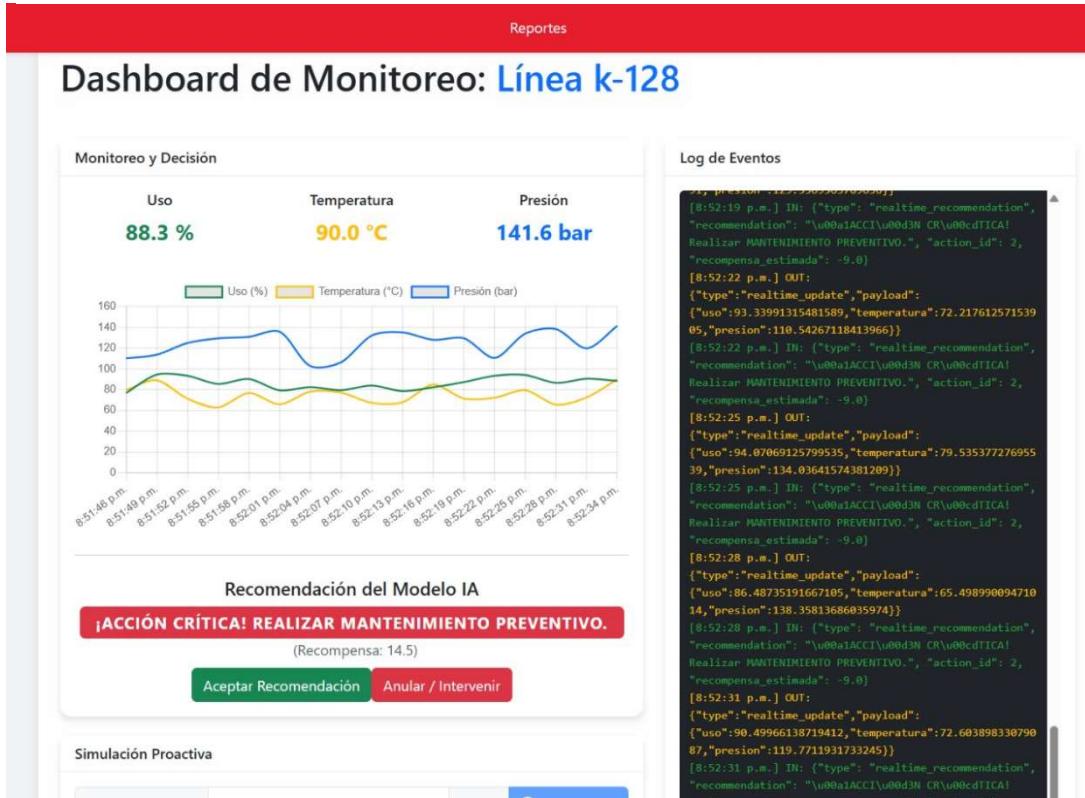
Fuente: Elaboración propia 2025

El entorno, construido sobre la librería *Gymnasium*, define los límites de operación de los sensores como espacio de observación, mientras que el espacio de acciones se reduce a decisiones binarias: intervenir con mantenimiento o continuar en operación. El agente avanza paso a paso, evaluando datos históricos simulados para aprender una política óptima que maximice la disponibilidad del equipo y minimice tanto el

desgaste como los costos por intervenciones innecesarias. La función de recompensa penaliza las decisiones que llevan al fallo del equipo y premia aquellas que permiten mantenerlo dentro de condiciones seguras sin sobreactuar.

7.4. GENERAR REPORTES AUTOMATIZADOS DEL ESTADO DE LOS EQUIPOS EN FUNCIÓN DE LA INFORMACIÓN REGISTRADA

Figura 12: Interfaz de reporte del monitoreo de la Línea K-128



Fuente: Elaboración propia 2025

Se desarrolló un módulo que recopila y presenta información relevante sobre cada línea de embotellado en una interfaz clara y funcional. El sistema toma como base las variables registradas y genera un reporte dinámico que incluye, además, una recomendación emitida por el módulo de inteligencia artificial. La interfaz también incorpora un log de eventos que muestra cronológicamente las acciones ejecutadas sobre el equipo, como mantenimientos realizados, cambios de estado o alertas generadas, lo que permite una trazabilidad completa. Todo el proceso de generación

del reporte está automatizado, de modo que se actualiza de forma periódica o ante eventos significativos, garantizando que la información esté disponible para la toma de decisiones sin intervención manual.

7.5. DISEÑAR LA ARQUITECTURA DE RED Y CONSTRUIR EL MODELO DE CABLEADO ESTRUCTURADO EMPLEADO EN LA PLANTA

Para el diseño de la arquitectura de red se tomarán en cuenta todas las sucursales de Embol S.A. para desarrollar una red completa de IPv4.

Figura 13. Mapa de todas las sucursales de Embol S.A simulado en Packet Tracer



Fuente: Elaboración propia 2025

El diseño considera todas las sucursales operativas de la empresa, permitiendo una integración completa bajo una red lógica uniforme. Como punto de partida, se utilizará la dirección base 34.199.34.0, obtenida a partir de la dirección de la página web de Embol S.A., a partir de la cual se dividirá la red en cuatro subredes principales, cada una configurada para contener internamente más de 30 subredes lógicas, suficientes para distribuir direcciones a todos los PCs, PLCs y dispositivos de control instalados en las líneas de embotellado.

7.5.1. División de subredes

Para obtener una dirección IP no muy alejada a la realidad, se realizó ping a la página de Embol S.A. donde se obtuvo la siguiente IP (figura 9):

Figura 14. Ping a embol.com

```
Haciendo ping a embol.com [34.199.34.72] con 32 bytes de datos:  
Tiempo de espera agotado para esta solicitud.  
  
Estadísticas de ping para 34.199.34.72:  
Paquetes: enviados = 4, recibidos = 0, perdidos = 4  
(100% perdidos),
```

Fuente: Elaboración propia 2025

A raíz de esta IP “34.199.34.72” se realizó la operación and (ANEXO C) para obtener el nombre de red “34.199.34.0”, con una máscara de red de prefijo /25 para que cada sucursal cuente con 128 hosts para las PCs, cada sucursal obteniendo las siguientes subredes:

Sucursal 1 – “Santa Cruz”:

- *Dirección de Red: 34.199.34.0/25*
- *Máscara de subred: 255.255.255.128*
- *Rango de Hosts: 34.199.34.1 a 34.199.34.126*
- *Dirección de Broadcast: 34.199.34.127*

Sucursal 2 – “La Paz”:

- *Dirección de Red: 34.199.34.128/25*
- *Máscara de subred: 255.255.255.128*
- *Rango de Hosts: 34.199.34.129 a 34.199.34.254*
- *Dirección de Broadcast: 34.199.34.255*

Sucursal 3 – “Cochabamba”:

- *Dirección de Red: 34.199.35.0/25*
- *Máscara de subred: 255.255.255.128*
- *Rango de Hosts: 34.199.35.1 a 34.199.35.126*
- *Dirección de Broadcast: 34.199.35.127*

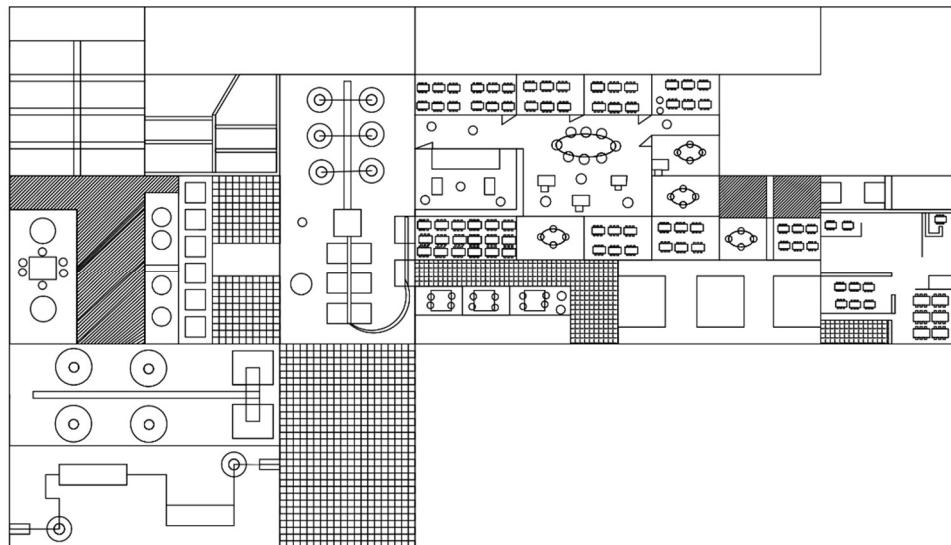
Sucursal 4 – “Tarija”:

- *Dirección de Red: 34.199.35.128/25*
- *Máscara de subred: 255.255.255.128*
- *Rango de Hosts: 34.199.35.129 a 34.199.35.254*
- *Dirección de Broadcast: 34.199.35.255*

7.5.2. Cableado estructurado

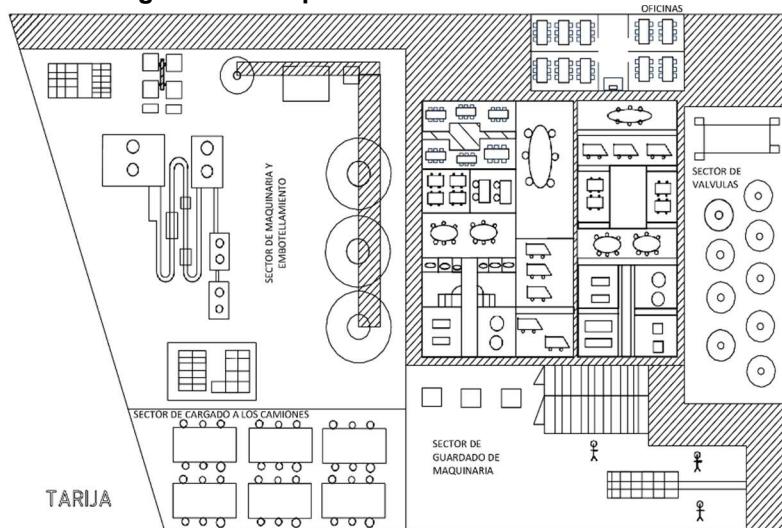
Para el cableado estructurado primero se realizo los mapas de cada sucursal para posteriormente subirlo a Packet Tracer y utilizarlos como base para el cableado de cada sucursal.

Figura 15. Croquis de la sucursal de LA PAZ



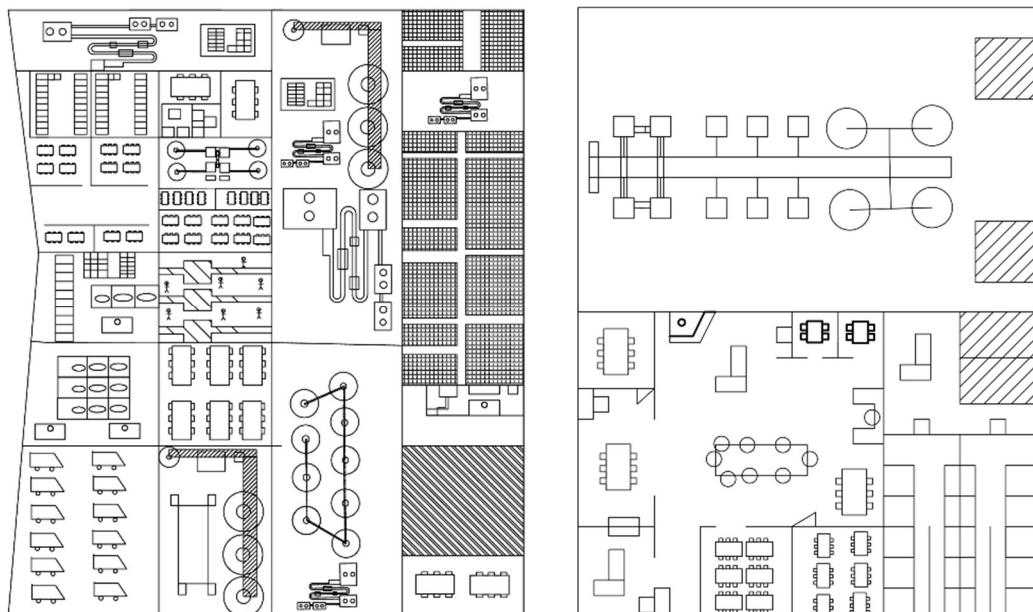
Fuente: Elaboración propia 2025

Figura 16. Croquis de la sucursal de TARIJA



Fuente: Elaboración propia 2025

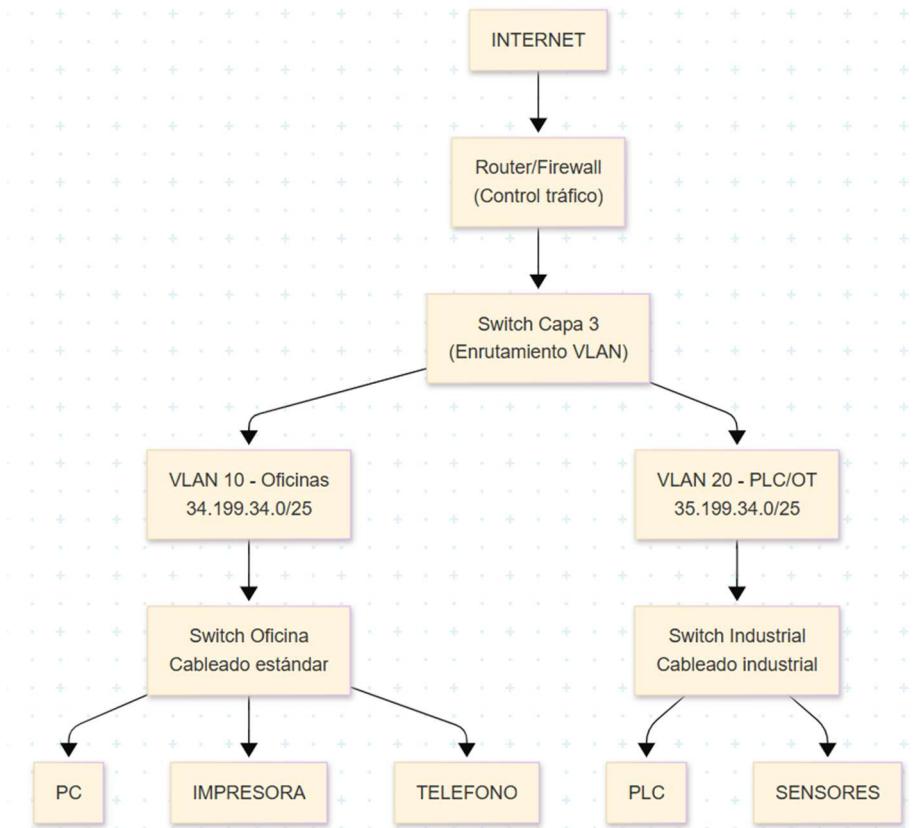
Figura 17. Croquis de las sucursales de SANTA CRUZ y COCHABAMBA



Fuente: Elaboración propia 2025

Después de la graficación de los mapas se realizó el cableado estructura donde se escogió el siguiente modelo para la división de la red de cada sucursal:

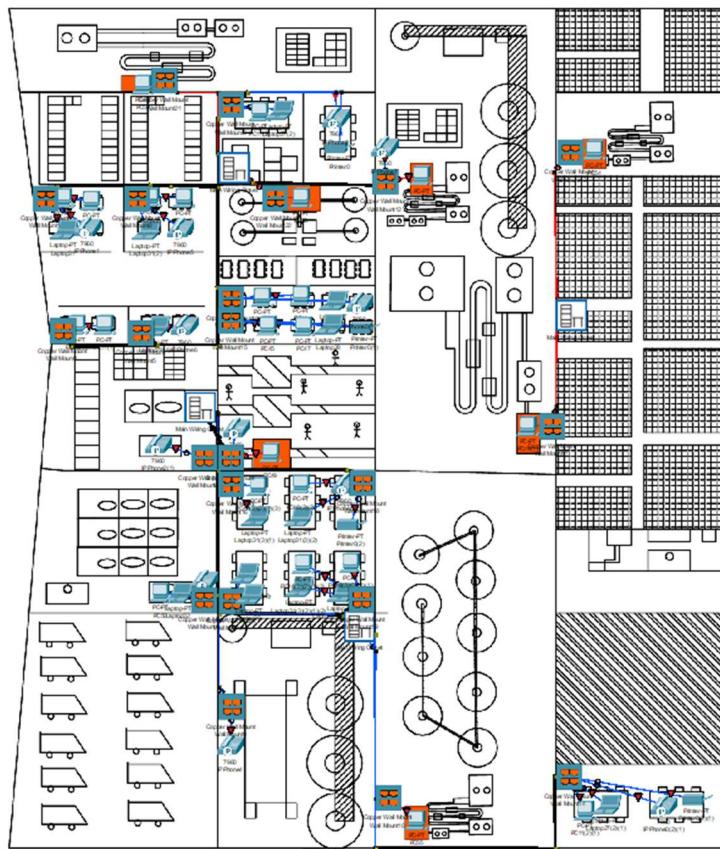
Figura 18: Diagrama de la división del cableado estructurado de cada sucursal



Fuente: Elaboración propia 2025.

Este diagrama refleja la organización jerárquica y funcional de la red, el cual fue implementado y documentado en Packet Tracer. El diseño parte de la conexión a Internet, que es gestionada por un router encargado del control de tráfico entrante y saliente. A partir de este, se conecta un switch de capa 3 que actúa como núcleo central, permitiendo el enrutamiento entre VLANs definidas para separar los entornos de oficina (VLAN 10) y de operación/PLC (VLAN 20), cada una con su propio rango de subred y color codificado. Desde el núcleo, la red se divide en dos ramas principales: una orientada a oficinas, que se conecta a través de un switch de acceso tradicional con cableado estándar, y otra hacia el entorno industrial, conectada a través de un switch industrial con cableado reforzado. En la VLAN de oficina se conectan dispositivos como PCs, teléfonos IP e impresoras, mientras que en la VLAN industrial se conectan equipos de control como PLCs y sensores.

Figura 19: Imagen del cableado estructurado realizado en la planta de Santa Cruz



Fuente: Elaboración propia 2025.

El cableado estructurado fue desarrollado en Cisco Packet Tracer con base en el diseño lógico de red planteado para la planta. Se implementó una arquitectura jerárquica compuesta por niveles de acceso, distribución y núcleo, empleando cableado acorde al entorno donde se encuentra cada segmento. Para la red de oficinas (VLAN 10), se utilizó cableado UTP categoría 6, conectado a través de patch panels y canalizaciones organizadas que parten desde el switch de acceso hasta cada estación de trabajo, teléfonos IP e impresoras, garantizando una velocidad de transmisión adecuada y una instalación limpia y mantenible. En el entorno industrial (VLAN 20), se aplicó cableado blindado (STP o industrial-grade) que va desde el switch industrial hacia los PLCs y sensores. Además, se respetaron normas de distancia máxima entre puntos, etiquetado de puertos, y separación física entre canalizaciones eléctricas y de datos.

7.6. DESARROLLAR UN SERVIDOR BÁSICO PARA ALOJAR, EJECUTAR Y RESPALDAR EL SISTEMA DE MONITOREO Y CONTROL BASADO EN MODELOS DE INTELIGENCIA ARTIFICIAL Y CADENAS DE MÁRKOV.

7.6.1. Definir las especificaciones esenciales para implementar un servidor básico que soporte la ejecución y almacenamiento del sistema desarrollado.

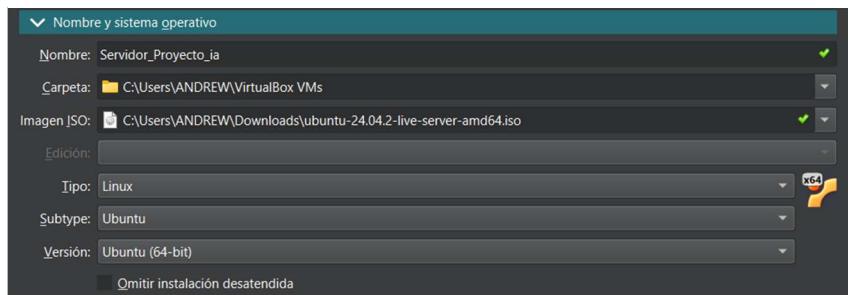
Se debe implementar un servidor que pueda almacenar el programa el cual los dispositivos deberán estar conectados de forma local en la misma red para poder acceder a este programa, esto lo lograremos haciendo un deployment del programa dentro de un servidor local.

7.6.2. Seleccionar la arquitectura de software y hardware adecuada para la implementación del servidor (local o virtualizado).

Necesitamos un equipo con la capacidad adecuada para ejecutar nuestro software desarrollado; para fines de demostración, se utilizará Oracle VirtualBox versión 7.1.8.

Al mismo tiempo usaremos un ISO del sistema operativo Ubuntu Server versión del sistema operativo Ubuntu diseñada específicamente para funcionar como servidor. A diferencia de la edición de escritorio, Ubuntu Server no incluye una interfaz gráfica por defecto, ya que está optimizado para el rendimiento, la seguridad y la administración remota a través de la línea de comandos.

Figura 20: Creación del servidor con especificaciones



Fuente: Elaboración propia 2025

Se crea la máquina virtual con las siguientes configuraciones, las cuales pueden ampliarse según los requerimientos del servidor.

Debemos seguir los pasos cuidadosamente al momento de instalar el sistema operativo dentro de la máquina virtual ya que algún error podría alterar el funcionamiento del servidor.

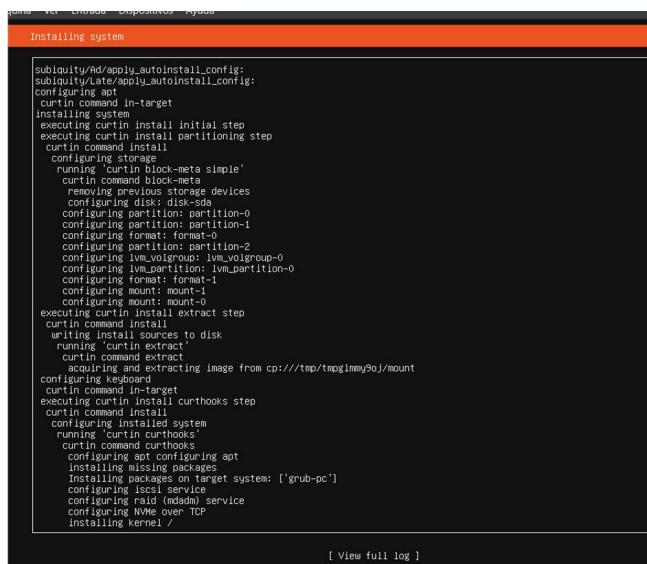
Figura 21: Inicio de instalación de la máquina virtual



Fuente: Elaboración propia 2025

Se sabrá que la instalación esta lista cuando tenemos el siguiente mensaje:

Figura 22: Final de la instalación de la máquina virtual



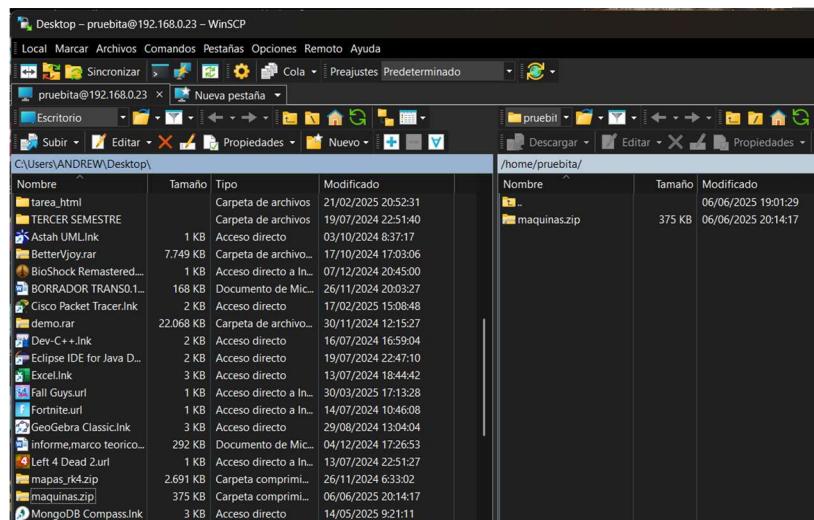
Fuente: Elaboración propia 2025

La máquina virtual con Ubuntu Server quedó configurada y lista para su uso; únicamente se requiere ingresar el usuario y la contraseña definidos durante la instalación del sistema operativo.

7.6.3. Configurar el entorno del servidor con los servicios necesario.

Se debe averiguar con que dirección ip está el servidor para poder acceder desde otros equipos. Posterior a esto se debe configurar el router donde estará conectado el servidor y los equipos, esta configuración es variable a la marca y modelo del router, debemos abrir puertos para que accedan los dispositivos. El siguiente paso es opcional: se puede configurar el servidor para que los cambios en la dirección IP no afecten su funcionamiento, lo cual realizaremos para mayor estabilidad y continuidad en el acceso. A continuación, se procede con la instalación de los componentes necesarios para el funcionamiento del servidor. Primero, se instala el servidor web Apache, que permitirá alojar y servir la aplicación desarrollada. Luego, se configura el gestor de bases de datos PostgreSQL, encargado de almacenar y administrar la información utilizada por el sistema. Finalmente, se instala el servicio SSH, el cual facilitará la transferencia segura del programa y otros archivos necesarios hacia la máquina virtual. Transferimos el zip de nuestro programa con el programa WinSCP.

Figura 23: Transferencia del sistema comprimido en un zip

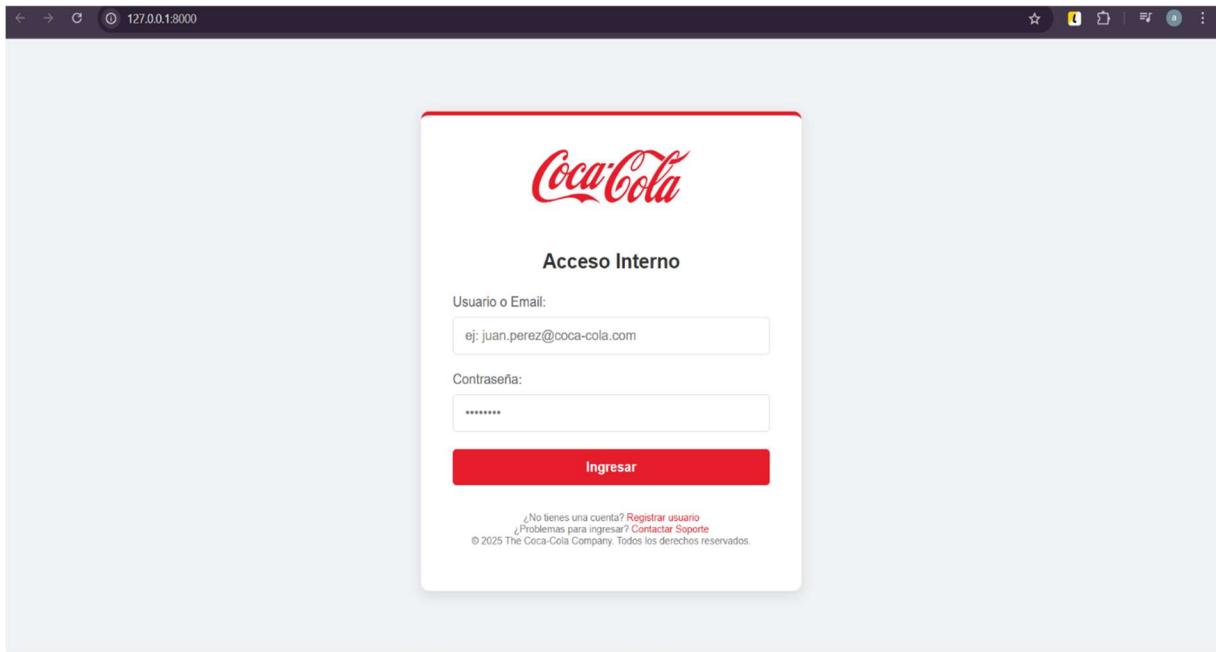


Fuente: Elaboración propia 2025

Ya en el servidor descomprimimos el programa con el comando unzip e instalar las librerías requeridas ya podemos ejecutar el sistema. Solamente es colocar la ip y el puerto donde está corriendo el programa y cualquier dispositivo puede ingresar.

7.6.4. Probar la estabilidad, conectividad y rendimiento del servidor con la carga real del sistema.

Figura 24: Interfaz de ingreso al sistema desde el servidor



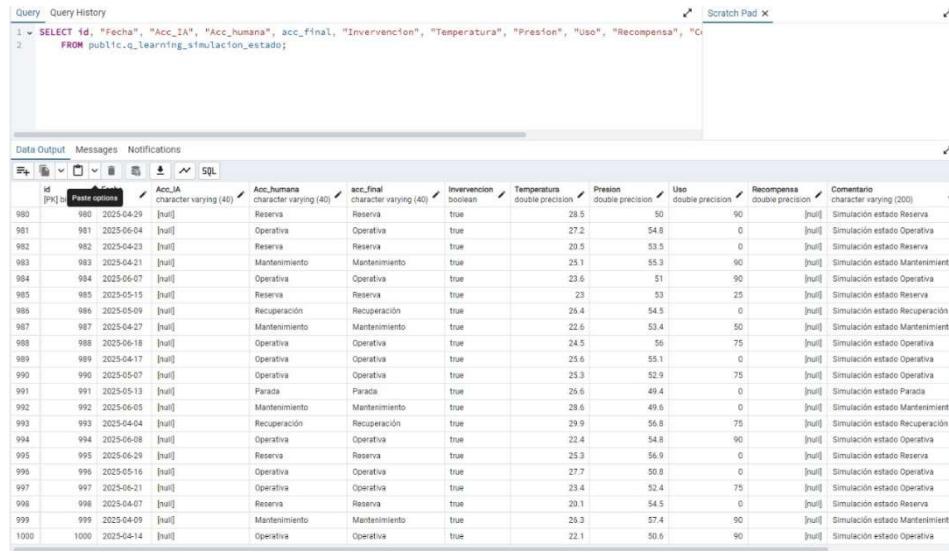
Fuente: Elaboración propia 2025

El servidor permite el almacenamiento y la recuperación de datos desde cualquier dispositivo conectado a la misma red local, garantizando el acceso dentro del entorno de red compartido.

7.7. REALIZAR PRUEBAS FUNCIONALES AL SISTEMA COMPLETO INTEGRADO

Se realizaron pruebas funcionales e integradas al sistema completo, enfocadas en la interacción entre el modelo de Márkov y el módulo de Inteligencia Artificial (IA) bajo condiciones simuladas.

Figura 25: Datos de condiciones simuladas.



The screenshot shows a database interface with a SQL query editor at the top and a data preview table below. The query is:

```

Query Query History
1. SELECT id, "Fecha", "Acc_IA", "Acc_humana", acc_final, "Inverencion", "Temperatura", "Presion", "Uso", "Recompensa", "C
2.      FROM public.q_learning_simulacion_estado;
  
```

The data table has the following columns and rows:

	Id [Pk]	Creado	Acc_IA	Acc_humana	acc_final	Inverencion	Temperatura	Presion	Uso	Recompensa	Comentario
980	980	2025-04-29	[null]	Reserva	true		28.5	50	90	[null]	Simulación estado Reserva
981	981	2025-04-04	[null]	Operativa	true		27.2	54.8	0	[null]	Simulación estado Operativa
982	982	2025-04-23	[null]	Reserva	true		20.5	53.5	0	[null]	Simulación estado Reserva
983	983	2025-04-21	[null]	Mantenimiento	true		25.1	55.3	90	[null]	Simulación estado Mantenimiento
984	984	2025-06-07	[null]	Operativa	true		23.6	51	90	[null]	Simulación estado Operativa
985	985	2025-05-15	[null]	Reserva	true		23	53	25	[null]	Simulación estado Reserva
986	986	2025-04-09	[null]	Recuperación	true		26.4	54.5	0	[null]	Simulación estado Recuperación
987	987	2025-04-27	[null]	Mantenimiento	true		22.6	53.4	50	[null]	Simulación estado Mantenimiento
988	988	2025-06-18	[null]	Operativa	true		24.5	56	75	[null]	Simulación estado Operativa
989	989	2025-04-17	[null]	Operativa	true		25.6	55.1	0	[null]	Simulación estado Operativa
990	990	2025-05-07	[null]	Operativa	true		23.3	52.9	75	[null]	Simulación estado Operativa
991	991	2025-05-13	[null]	Parada	true		26.5	49.4	0	[null]	Simulación estado Parada
992	992	2025-06-05	[null]	Mantenimiento	true		28.6	49.6	0	[null]	Simulación estado Mantenimiento
993	993	2025-04-04	[null]	Recuperación	true		29.9	56.8	75	[null]	Simulación estado Recuperación
994	994	2025-06-08	[null]	Operativa	true		22.4	54.8	90	[null]	Simulación estado Operativa
995	995	2025-06-29	[null]	Reserva	true		25.3	56.9	0	[null]	Simulación estado Reserva
996	996	2025-05-16	[null]	Operativa	true		27.7	50.8	0	[null]	Simulación estado Operativa
997	997	2025-06-21	[null]	Operativa	true		23.4	52.4	75	[null]	Simulación estado Operativa
998	998	2025-04-07	[null]	Reserva	true		20.1	54.5	0	[null]	Simulación estado Reserva
999	999	2025-04-09	[null]	Mantenimiento	true		26.3	57.4	90	[null]	Simulación estado Mantenimiento
1000	1000	2025-04-14	[null]	Operativa	true		22.1	50.6	90	[null]	Simulación estado Operativa

Fuente: Elaboración propia 2025.

Durante las pruebas, se verificó la estabilidad y correcto funcionamiento de la conexión WebSocket, así como la comunicación bidireccional entre módulos. El sistema respondió adecuadamente con recomendaciones en tiempo real, indicando acciones operativas normales y alertas críticas para mantenimiento preventivo según los datos de uso, temperatura y presión enviados. Aunque se detectaron algunos mensajes de tipo desconocido, no se presentaron fallos críticos que comprometieran la operación. En general, los resultados evidencian que los módulos están correctamente integrados y funcionan coordinadamente dentro del sistema, cumpliendo los objetivos planteados, aunque se recomienda mejorar el manejo de mensajes no previstos para futuras iteraciones.

Figura 26: Imagen de la compilación e integración de los resultados del sistema

```
192.168.1.7:55785 - - [06/Jun/2025:20:51:43] "WCONNECT /ws/maquina/L%C3%ADnea%20k-128/" - -
WebSocket conectado para la línea: Línea k-128 por el usuario: admin
ERROR en WebSocket (Línea k-128): Tipo de mensaje desconocido: simulate_future
WebSocket conectado para la línea: Línea k-128 por el usuario: admin
ERROR en WebSocket (Línea k-128): Tipo de mensaje desconocido: simulate_future
ERROR en WebSocket (Línea k-128): Tipo de mensaje desconocido: simulate_future
192.168.1.7:57947 - - [06/Jun/2025:20:53:11] "GET /qlearning/dashboard/L%C3%ADnea%20k-128/" 200 15411
192.168.1.7:55785 - - [06/Jun/2025:20:53:11] "WDISCONNECT /ws/maquina/L%C3%ADnea%20k-128/" - -
WebSocket desconectado de la línea: Línea k-128
192.168.1.7:57954 - - [06/Jun/2025:20:53:11] "WSCONNECTING /ws/maquina/L%C3%ADnea%20k-128/" - -
192.168.1.7:57954 - - [06/Jun/2025:20:53:11] "WCONNECT /ws/maquina/L%C3%ADnea%20k-128/" - -
WebSocket conectado para la línea: Línea k-128 por el usuario: admin
[]

--> DATO ENVIADO (t=426): Uso=54.2%, Temp=58.8°C, Pres=78.3bar
<-- MENSAJE RECIBIDO: {'type': 'realtime_recommendation', 'recommendation': 'Acción sugerida: Forzar a estado OPERATIVA.', 'action_id': 0, 'recompensa_estimada': 1.0}
-> DATO ENVIADO (t=427): Uso=57.4%, Temp=60.5°C, Pres=74.4bar
<-- MENSAJE RECIBIDO: {'type': 'realtime_recommendation', 'recommendation': 'ACCIÓN CRÍTICA! Realizar MANTENIMIENTO PREVENTIVO.', 'action_id': 2, 'recompensa_estimada': -9.0}
-> DATO ENVIADO (t=428): Uso=52.1%, Temp=57.2°C, Pres=67.8bar
<-- MENSAJE RECIBIDO: {'type': 'realtime_recommendation', 'recommendation': 'Acción sugerida: Forzar a estado OPERATIVA.', 'action_id': 0, 'recompensa_estimada': 1.0}
-> DATO ENVIADO (t=429): Uso=55.1%, Temp=60.5°C, Pres=68.6bar
<-- MENSAJE RECIBIDO: {'type': 'realtime_recommendation', 'recommendation': 'Acción sugerida: Forzar a estado OPERATIVA.', 'action_id': 0, 'recompensa_estimada': 1.0}
```

Fuente: Elaboración propia 2025.

8. CONCLUSIONES Y RECOMENDACIONES.

8.1. CONCLUSIONES.

- Se logró un análisis exhaustivo de la información técnica y operativa proveniente de los sistemas de automatización (PLC), así como de los registros históricos de mantenimiento y calidad, permitiendo identificar las variables clave que afectan el comportamiento de los equipos. Esta información fue fundamental para construir un modelo predictivo preciso y alineado con la realidad operativa de las líneas de embotellado.
- El modelo probabilístico basado en Cadenas de Márkov fue correctamente diseñado e implementado, representando de manera efectiva las posibles transiciones de estado de los equipos. Este modelo se constituyó como el núcleo predictivo del sistema, proporcionando una base sólida y confiable para anticipar el comportamiento operativo de las líneas.
- Se desarrolló y entrenó un módulo de inteligencia artificial capaz de detectar patrones no lineales y adaptarse a condiciones operativas variables. Su integración con el modelo estocástico permitió enriquecer las predicciones del sistema.

- El sistema generó de manera efectiva reportes automatizados con información actualizada sobre el estado de los equipos, incluyendo etiquetas de estado y alertas oportunas. Esta funcionalidad permitió fortalecer la toma de decisiones en mantenimiento, anticipando fallas y promoviendo acciones preventivas basadas en datos objetivos.
- Se diseñó e implementó con éxito una arquitectura de red robusta y un modelo de cableado estructurado que garantizó la conectividad y trazabilidad de todos los componentes del sistema. Este diseño permitió la integración eficiente de PLCs y administrativos, asegurando una comunicación estable entre los módulos de monitoreo y control.
- Se desarrolló un servidor funcional que aloja y respalda el sistema, permitiendo su ejecución continua y estable. Este servidor actúa como núcleo operativo donde convergen los datos de proceso, los modelos predictivos y las funciones de interfaz para usuarios y técnicos.
- Las pruebas funcionales confirmaron el correcto funcionamiento e integración de todos los módulos del sistema. Se validó la comunicación efectiva entre el modelo de Márkov y el módulo de IA, así como la correcta recepción, procesamiento y respuesta a datos simulados.

8.2. RECOMENDACIONES.

Se recomienda digitalizar por completo todos los registros operativos y de mantenimiento, incluyendo las paradas menores, para asegurar que el sistema predictivo opere con datos fiables y en tiempo real. La implementación del sistema debe iniciarse en las líneas de producción más críticas, como la K-128 y la Combi-Sidel 135, evaluando el rendimiento del modelo y corrigiendo posibles desviaciones antes de expandirlo al resto de la planta. Es fundamental capacitar al personal técnico en el uso del sistema, en la interpretación de alertas predictivas y en el mantenimiento de la infraestructura de red. Además, se recomienda establecer rutinas periódicas de reentrenamiento del módulo de IA con datos nuevos para asegurar que el sistema se mantenga actualizado y eficiente frente a cambios en las condiciones operativas.

9. BIBLIOGRAFÍA

- Baldivieso, G. V. (2016). *Balance de masa de tratamiento de agua para el consumo de la gestión 2017 de EMBOL S.A.* UNIVERSIDAD AUTONOMA. Obtenido de <https://es.scribd.com/document/402715092/practica-EMBOL-ULTIMISIMO-1-docx>
- Bendoraitis, A. (16 de marzo de 2025). *Daphne*. Obtenido de pybazaar.com: <https://www.pybazaar.com/resources/6EoUndvfR4Qx/>
- Bigelow, S. J. (19 de Julio de 2024). *¿Qué es NumPy? Explicación de su funcionamiento en Python*. Obtenido de TechTarget.com: <https://www.techtarget.com/whatis/definition/What-is-NumPy-Explaining-how-it-works-in-Python>
- Carnes, B. (21 de Marzo de 2023). *Utilice el gimnasio para el aprendizaje de refuerzo*. Obtenido de FreeCodeCamp.org: <https://www.freecodecamp.org/news/use-openai-gymnasium-for-reinforcement-learning/>
- Diaconu, A. (30 de abril de 2025). *WebSockets explicados*. Obtenido de ably.com: <https://ably.com/topic/websockets#:~:text=WebSocket%20es%20un%20protocolo%20de,el%20cliente%20y%20el%20servidor>.
- Doyle, K. (12 de Diciembre de 2024). *Django*. Obtenido de TecTarget.com: <https://www.techtarget.com/searchapparchitecture/tip/Django-vs-Flask-Comparing-Python-web-frameworks>
- Ferreira, J. (26 de Julio de 2022). *La librería pandas*. Obtenido de Aprendeconalf.es: <https://aprendeconalf.es/docencia/python/manual/pandas/>
- Forouzan, B. A. (2002). *Transmisión de Datos y Redes de Comunicaciones – 2da Edición*. España: McGraw-Hill.

Fuentes, G. (2012). *Bases de datos. Gestión.* Obtenido de Notas_del_curso_Bases_de_Datos: http://local.cua.uam.mx/pdfs/conoce/libroselec/Notas_del_curso_Bases_de_Datos.pdf

FUJITSU. (2023). El Mantenimiento Predictivo en la Industria 4.0. *Club Excelencia*, 37.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly.

GoDaddy, E. d. (27 de Diciembre de 2024). *Protocolos de red: Qué son?* Obtenido de godaddy.com : <https://www.godaddy.com/resources/es/tecnologia/que-son-los-protocolos-de-red-y-cuales-son-los-mas-importantes-hoy-en-dia>

González, A. C. (2024). Mantenimiento prescriptivo (RxM) Modelo de implementación. *CONGRESO DE MANTENIMIENTO & CONFIABILIDAD*, 39.

Goodfellow, I. B. (2016). *Deep Learning*. MIT Press.

Gutoswka, A. (3 de Julio de 2024). *¿Qué son los agentes de IA?* Obtenido de IBM: <https://www.ibm.com/es-es/think/topics/ai-agents>

Jackson, G. (15 de Febrero de 2024). *¿Qué es la topología de red?* Obtenido de IBM.com : <https://www.ibm.com/es-es/topics/network-topology#:~:text=La%20topolog%C3%ADA%20de%20red%20se,%2C%20conmutadores%2C%20repetidores%20y%20ordenadores.>

Joaquín Ortega Sánchez, Víctor Rivero Mercado. (2020). *Modelos Estocásticos I*. Cimat, A.C.

Joskowicz, J. (2006). *Cableado Estructurado*. Montevideo, Uruguay: Universidad de la Republica.

Pinzón, C. (2020). TIPOS DE MANTENIMIENTO. *Tipos de mantenimiento que pueden ser aplicados*, 17.

Raffin, A. (28 de Febrero de 2021). *Stable-Baselines3*. Obtenido de araffin.github.io:
<https://araffin.github.io/post/sb3/>

Raul Awati, Ben Lutkevich. (22 de Noviembre de 2024). *¿Qué es un framework?*
Obtenido de TechTarget.com:
<https://www.techtarget.com/whatis/definition/framework>

Rioja, R. M. (2025). *El cableado estructurado*. Obtenido de adrformacion.com:
https://www.adrformacion.com/knowledge/administracion-de-sistemas/el_cableado_structurado_de_una_red_de_area_local.html

Rioja, U. I. (20 de Noviembre de 2023). *Que son las librerias en programación y para que sirven.* Obtenido de unir.net:
<https://www.unir.net/revista/ingenieria/librerias-programacion/>

Sanhueza, P. U. (Julio de 2012). *Cableado Estructurado para Ambientes industriales*.
Obtenido de ElectroIndustria.mvc :
<https://www.emb.cl/electroindustria/articulo.mvc?xid=1866&ni=cableado-estructurado-para--ambientes-industriales>

UNAM. (2 de Julio de 2021). *Fundamentos Teóricos*. Obtenido de ptolomeo.unam.mx:
chrome-extension://efaidnbmnnibpcajpcglclefindmkaj/http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/779/A4.pdf?sequence=4&isAllowed=y

Universidad de Murcia. (2021). Tema 6. En *Estimación de parámetros, validación de modelos y análisis de sensibilidad* (pág. 6).

Watkins, C. &. (1992). *Q-learning. Machine Learning*.

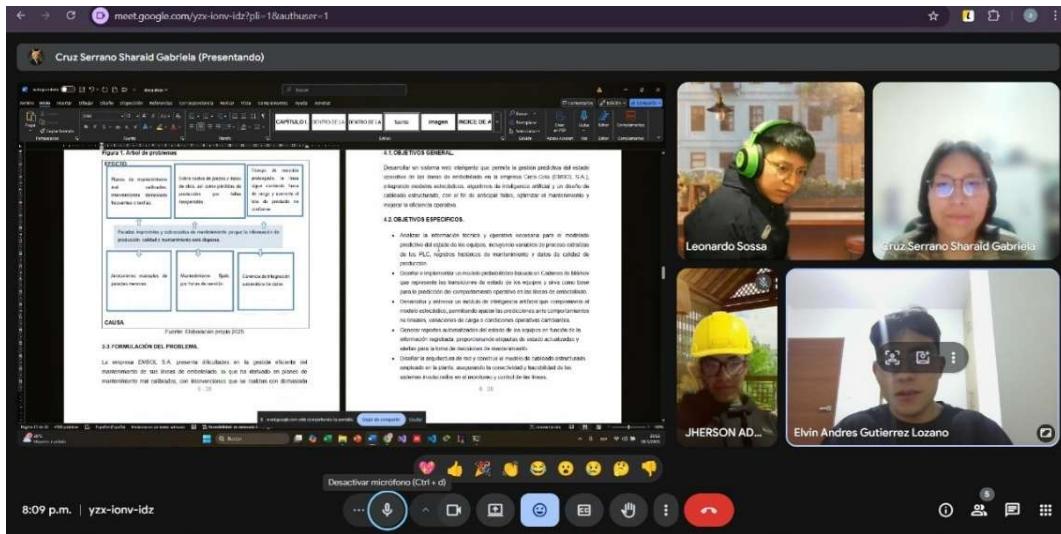
Watkins, C. &.-l. (2018). *Reinforcement Learning: An Introduction*. MIT Press.

ANEXOS

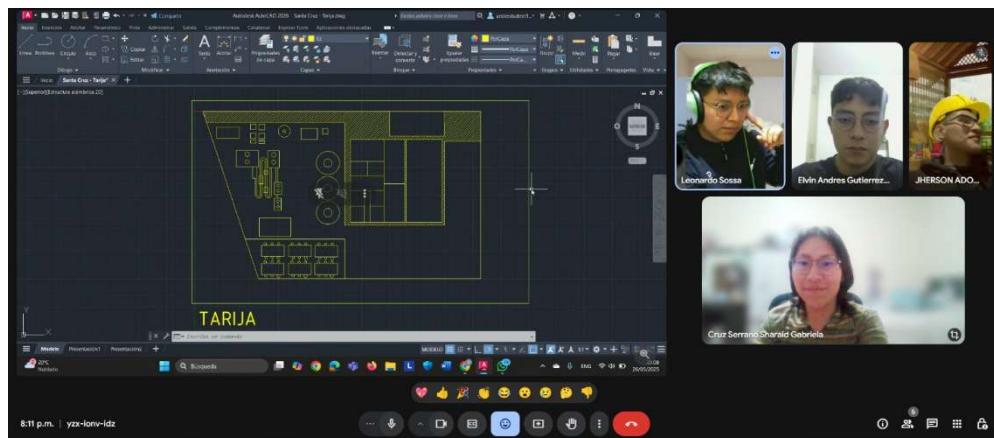


ANEXOS

Anexo A: REUNIÓN VIRTUAL (20/05/25)



Anexo B. REUNIÓN VIRTUAL (24/05/25)



Descripción: Realización de los mapas de cada sucursal.

ANEXO C: OPERACIÓN AND Y SUBDIVISIÓN DE LA RED

~~00100010. 11000111. 00100010. 01001000
11111111. 11111111. 11111110. 00000000
00100010. 11000111. 00100010. 00000000~~ → 122] 23 bits para red
9 bits para hosts

34. 199. 34. 0 → Nombre de la red

Subredes = La empresa embal s.a tiene 4 sucursales en Bolivia donde necesita que se divida su red en subredes.

$2^2 = 4$ subredes / máscara 23 bits para hosts

~~11111111. 11111111. 11111110. 00000000~~

4 bits para red

Con 7 bits para hosts significa que cada subred tendría $2^7 = 128$ direcciones totales de las cuales 126 serán utilizables, sin contar obviamente el broadcast y nombre de la red.

Subred - "cálculo"

Nombre de la red

1 primera subred

~~00100010. 11000111. 00100010. 00000000~~

34 . 199. 00100010. 00000000 → 1º

34 . 199. 00100010. 10000000 → 2º

34 . 199. 00100010. 01000000 → 3º

34 . 199. 00100010. 11000000 → 4º

1º-Santa Cruz = 34. 199. 34. 0 → Nombre de red

2º-La Paz = 34. 199. 34. 128 → Nombre de red

3º-Cochabamba = 34. 199. 34. 192 → Nombre de red

4º-Tarija = 34. 199. 34. 256 → Nombre de red

↓ Descripción detallada
atras

Descripción: Resolución de la operación de red para la obtención de nombre de red y cálculo de subdivisión de las redes para sucursal.